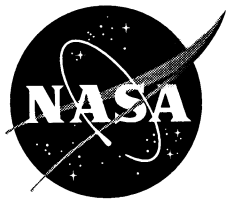
NASA/CR—2001-211389



Summary of ADTT Website Functionality and Features

Veronica Hawke, Trang Duong, Lawrence Liang, and Peter Gage

Prepared for NASA Ames Research Center
under Contract NAS2-00062 (Task 8)

December 2001

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at *http://www.sti.nasa.gov*

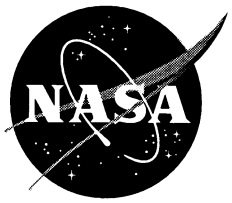
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/CR—2001-211389



Summary of ADTT Website Functionality and Features

Veronica Hawke, Trang Duong, Lawrence Liang, and Peter Gage
ELORET Corporation
690 West Fremont Ave., Suite 8
Sunnyvale, California 94087-4202

Prepared for NASA Ames Research Center
under Contract NAS2-00062 (Task 8)

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

December 2001

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

Table of Contents

Table of Contents (Cont.)

Table of Contents (Cont.)

Page intentionally left blank

1.0 Introduction

This report summarizes development of the ADTT web-based design environment by the ELORET team in 2000. The Advanced Design Technology Testbed had been in development for several years, with demonstration applications restricted to aerodynamic analyses of subsonic aircraft. The key changes achieved this year were improvements in Web-based accessibility, evaluation of collaborative visualization, remote invocation of geometry updates and performance analysis, and application to aerospace system analysis. Significant effort was also devoted to post-processing of data, chiefly through comparison of similar data for alternative vehicle concepts. Such comparison is an essential requirement for designers to make informed choices between alternatives.

The next section of this report provides more discussion of the goals for ADTT development. Section 3 provides screen shots from a sample session in the ADTT environment, including Login and navigation to the project of interest, data inspection, analysis execution and output evaluation. The following section provides discussion of implementation details and recommendations for future development of the software and information technologies that provide the key functionality of the ADTT system. Section 5 discusses the integration architecture for the system, which links machines running different operating systems and provides unified access to data stored in distributed locations. Security is a significant issue for this system, especially for remote access to NAS machines, so Section 6 discusses several architectural considerations with respect to security. Additional details of some aspects of ADTT development are included in Appendices.

2.0 Goals and Achievements

Integrated Design Systems efficiently connect CAD geometry, analysis software, existing data and hardware with the people who use these resources to assist in the design of engineering artifacts. ADTT is intended to accurately capture and apply multi-disciplinary and multi-physics expertise that is required to analyze aerospace vehicles. It should also facilitate the organization and evaluation of information generated in the design activity and it should provide methods to conveniently share this information with remotely-located team members.

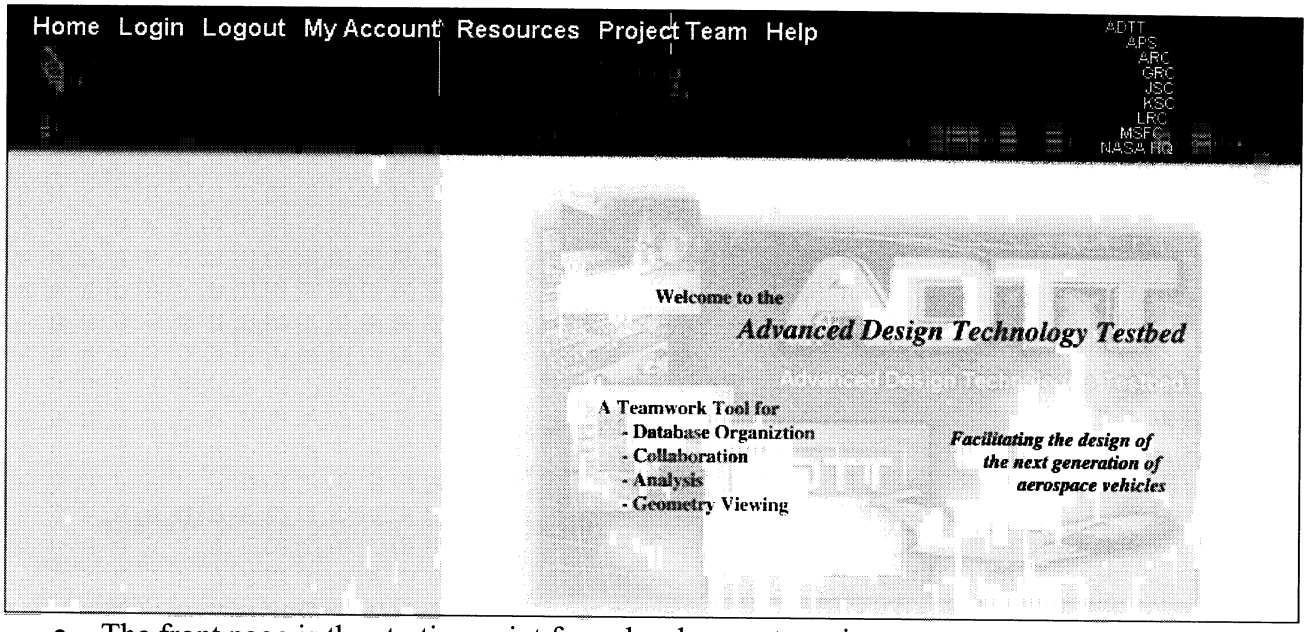
Design of Second Generation Reusable Launch Vehicles (Gen II RLV) was chosen as the aerospace application to be demonstrated in 2000. The Gen II Program Office had four generic concepts for evaluation: single stage Wing Body, single stage Lifting Body, two stage BiMese (identical lifting bodies, but much smaller than the single stage vehicle) and Shuttle-derived FlyBack Booster. Data for each of these vehicles were collected from civil servants who had run HAVOC analyses in support of the ISAT study and stored in the ADTT database. New ProEngineer models were constructed for each concept, and Catia parts were included for the FlyBack booster. Parametric design changes were also demonstrated for one concept (the Wing Body was chosen).

The Gen II Program Office was particularly interested in distributed teams having collaborative access to vehicle data. After a review of several commercial off the shelf collaborative environments, Vuent was selected for implementation in this proto-type. This system was implemented successfully, as reflected in the Sample Session in Section 3. Several operational deficiencies were identified and are reported fully in Appendix B. These weaknesses prompted an extensive evaluation of alternative 3D Viewing environments, and a report on that effort is included as Appendix A. Also included in Appendix A is a summary and general evaluation of the third party vendors/software both incorporated into the website and evaluated or considered along the way.

ADTT provides access to an object-oriented database through Java interface applications, and includes Unix scripting to drive distributed CAD and disciplinary analysis software. Development of the database and interfaces continued in the current task. The VUENT system uses a relational database (Microsoft Access) to store CAD-related data, and communication between ObjectStore and the relational database were successfully established. Difficulties with object-oriented database update were identified as the database grew: delays became unacceptably long. Alternative database technologies will need to be implemented for an operational system.

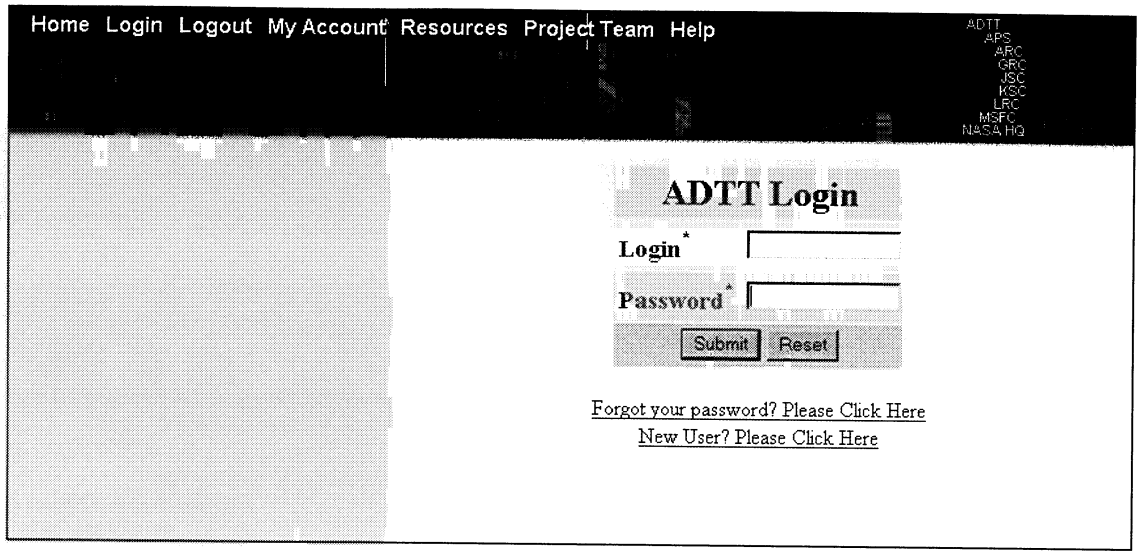
3.0 Sample Session

3.1 Front Page



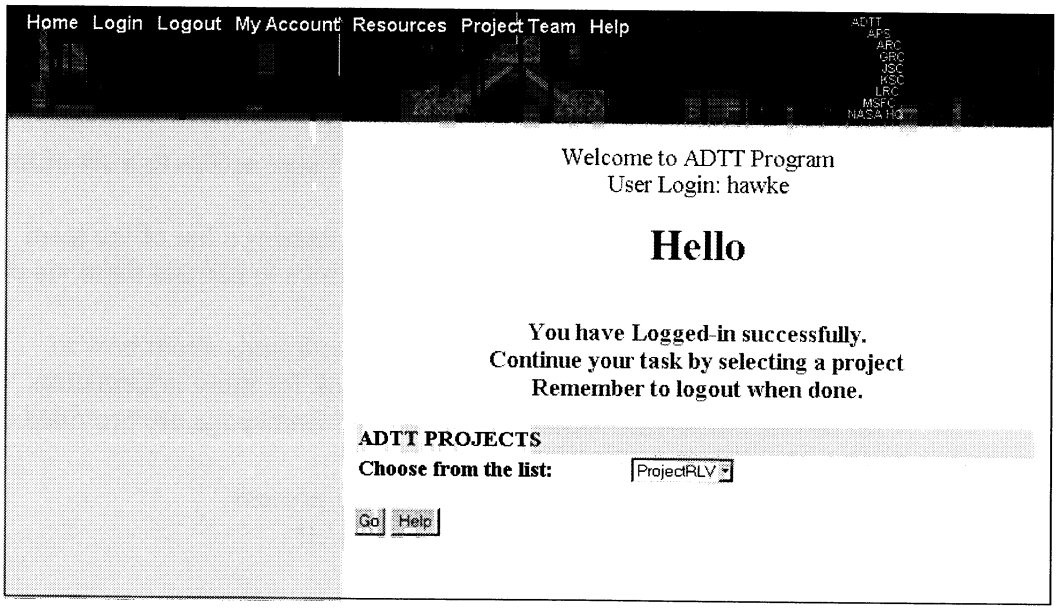
- The front page is the starting point for a development session.
- The top bar is available throughout the session and provides the user with a set of tools:
  - **Home** sends the user back to the front page of the site.
  - **Login** and **Logout** are self descriptive. More details are given below.
  - **My Account** allows users to change their passwords and update their account information.
  - **Resources** when functional will provide the user access to a set of tools such as a unit conversion calculator.
  - **Project Team** pops up a new window and contains a list of the team members working on the ADTT project and links to contact them via e-mail.
  - **Help** when functional will provide the user with hints about how to navigate through the website and where to look for more information.
  - A set of links to NASA websites and the APS homepage are provided on the right-hand side of the banner.
- The user will use **Login** to get to the next step in the process.

3.2 Login Form



- Provides a relatively secure starting point for entry in to the ADTT Website.
- Currently the passwords are not fully encrypted allowing the administrator access to the users passwords, so the users are asked to provide non-sensitive passwords at this time.

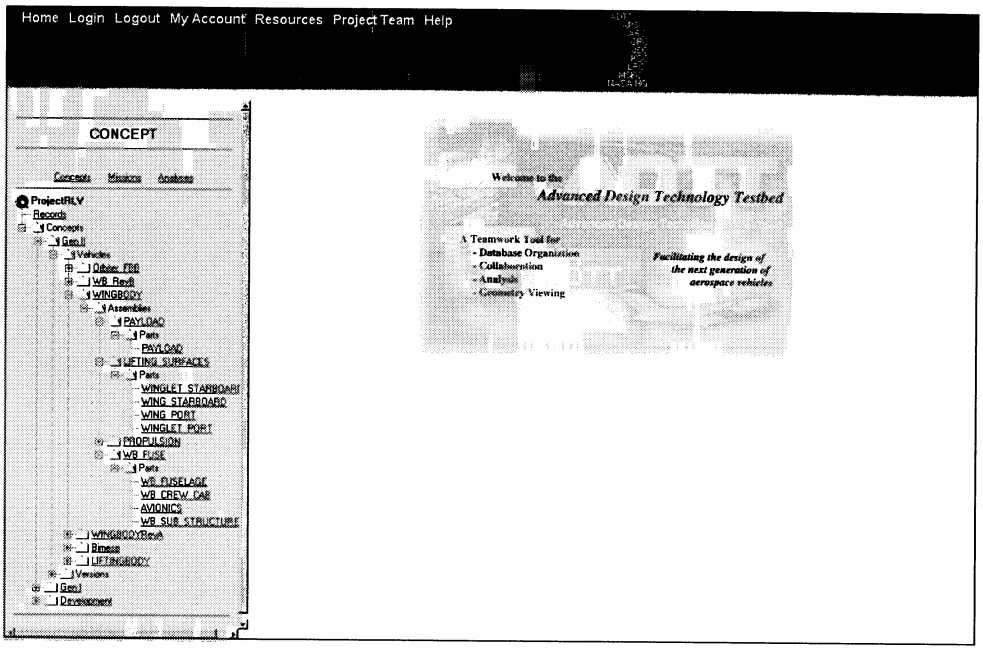
3.3 Project Selection Page



- After logging in the user selects the project to be worked on.
- At this time the only project available for selection is **Project RLV**. In the future additional projects may be added.

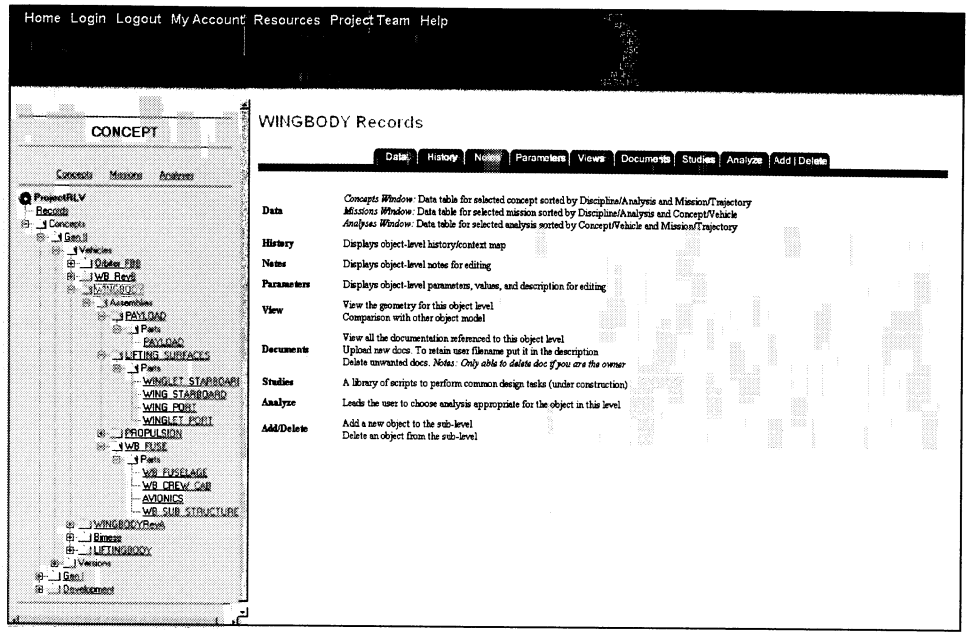
- The **Go** button is clicked to move to the next page.

3.4 Concept Tree



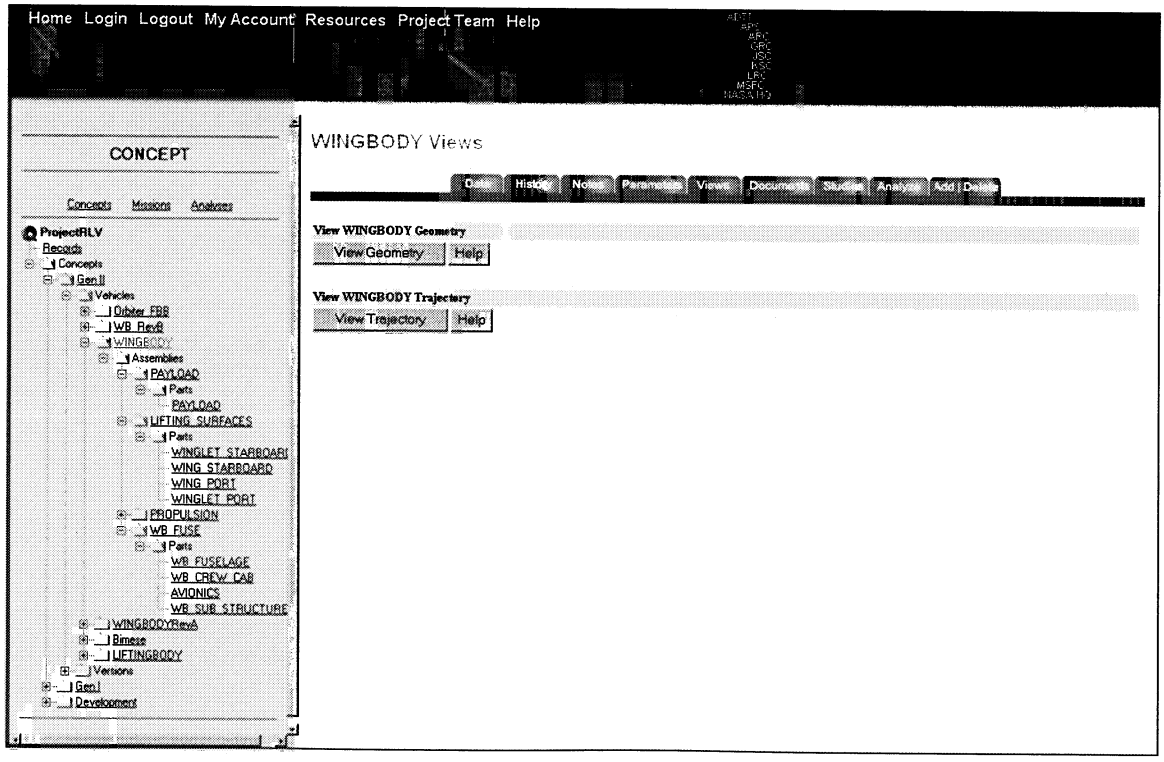
- The Concept Tree is the default starting point after the user selects a project. The other pages Missions and Analyses will be covered later.
- The Concept Tree organizes the available starting vehicles.
- The different levels of the tree allow the user access to the sub-assemblies and parts of the different vehicles.
- The next step a user may take is to select a baseline vehicle such as the **Wingbody**. Using the **Wingbody** link the user is moved to the Vehicle Records page.

3.5 Vehicle Records Page



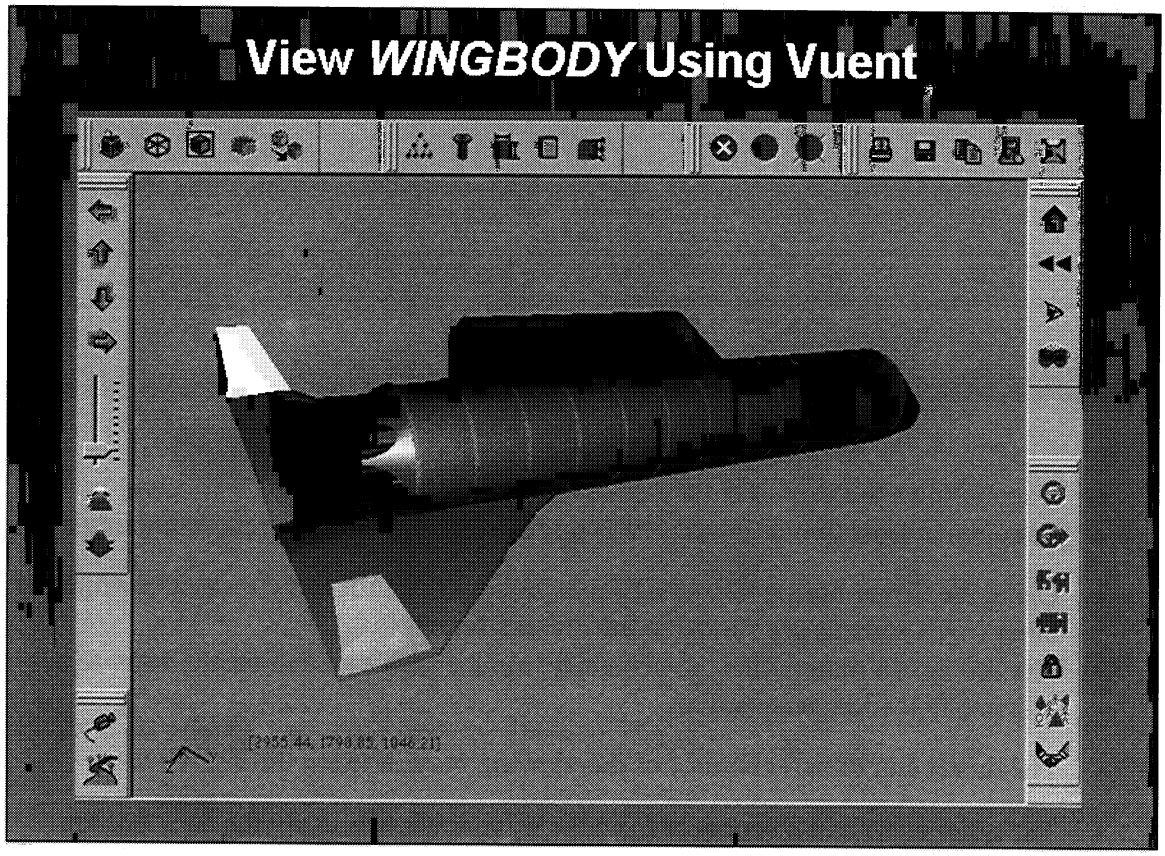
- A brief description of each tab's function is shown above in the figure.
- The user may want to view the geometry of the baseline model next. This process is started by selecting the **Views** tab.

3.6 Views Window



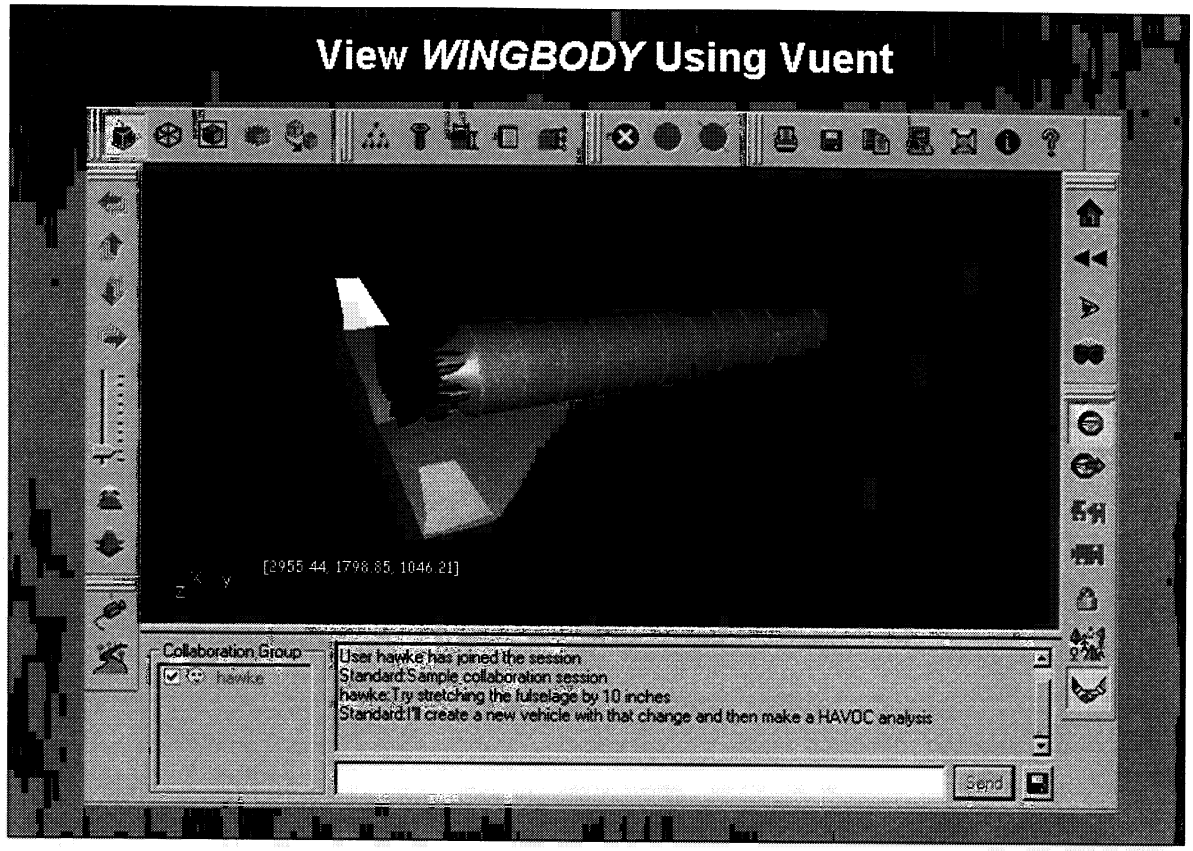
- The geometry of a vehicle can be examined in a 3D viewer via the **View Geometry** button.
- The **View Trajectory** currently shows a sample analysis solution on the LIFTINGBODY geometry and also shows a representative mission path. This tool is not yet fully developed but could provide a good means of solution viewing in the future.
- The two **Help** buttons contain some information that can help the user with known problems with the applications.
- Clicking the **View Geometry** button will bring up a separate window shown on the next page.

3.7 View Vehicle Geometry



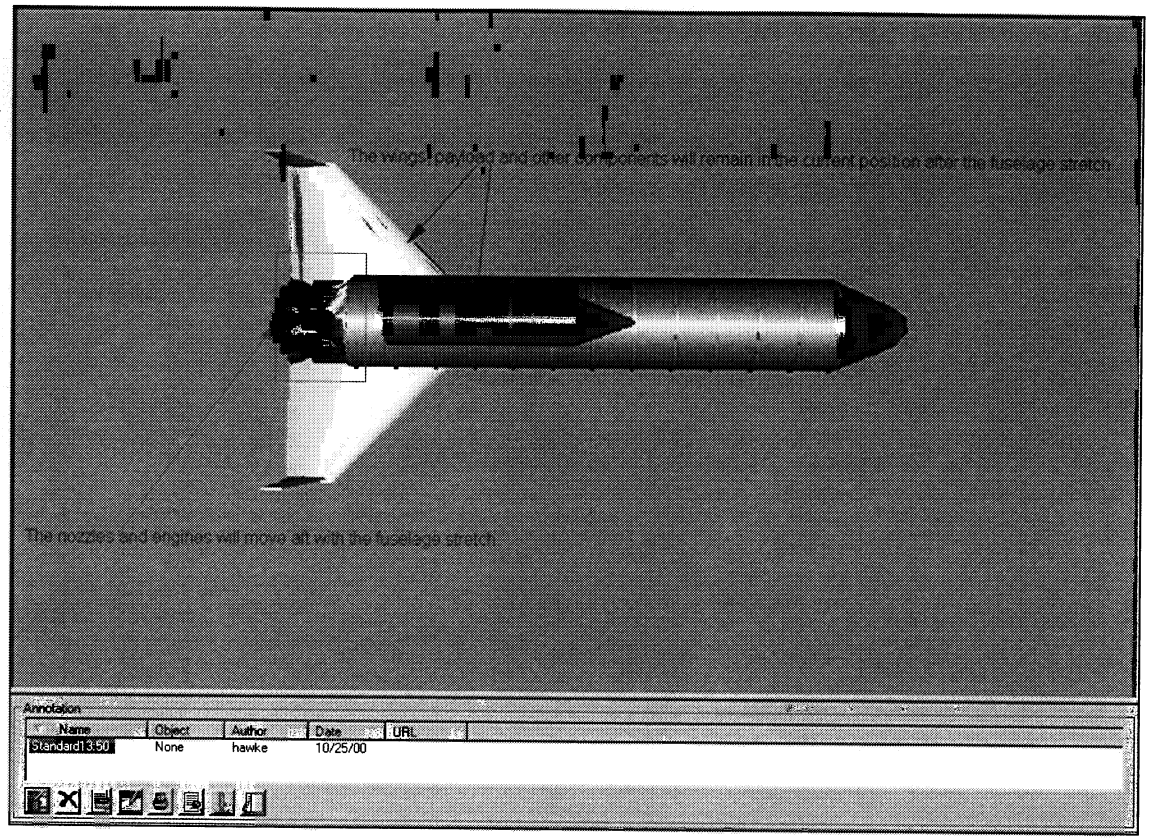
- Allows user to look at 3D geometry based on original CAD data.
- Full summary of this tool can be found in Reference 1.

3.8 Vuent Collaborative Session



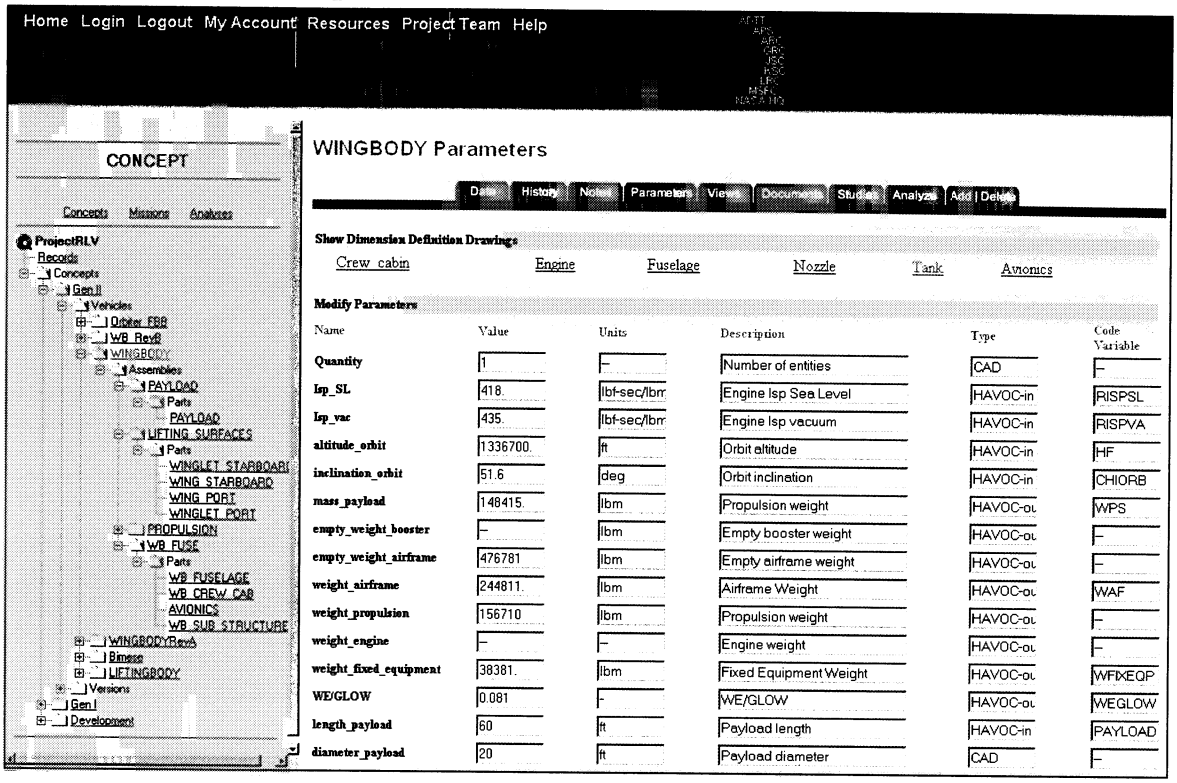
- Vuent allows the user to enter a collaborative session using the  button.
- Requires at least two users to be viewing the same geometry at the same time.
- Using the driver button  one of the users takes control of the session while the other user's screens are slaved to the driver's screen.
- The driver can manipulate the geometry, run the path tool and set the view to a previously loaded perspective.
- The driver can transfer control of the viewing session using the  button.
- The other users can disconnect from the driver's session using the  button.

3.9 Vuent Geometry Annotation



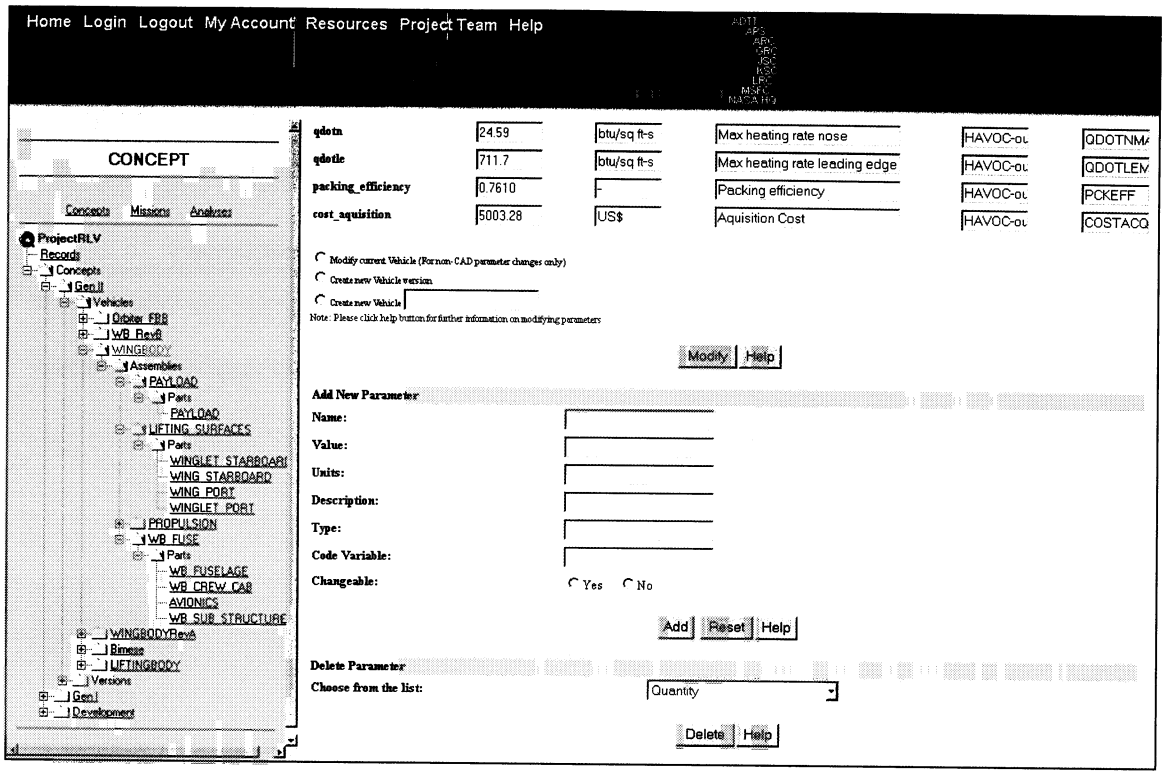
- After a collaborative session one of the users may provide some documentation of the upcoming change via an annotation by clicking the **Annotation** button  .
- Though this functionality can be useful there are some drawbacks that are described in detail in Reference 2.
- After deciding on the next change the user would move to the **CONCEPTS Parameters** tab to modify the current vehicle or create a new vehicle.

3.10 Parameters Top Portion of Window



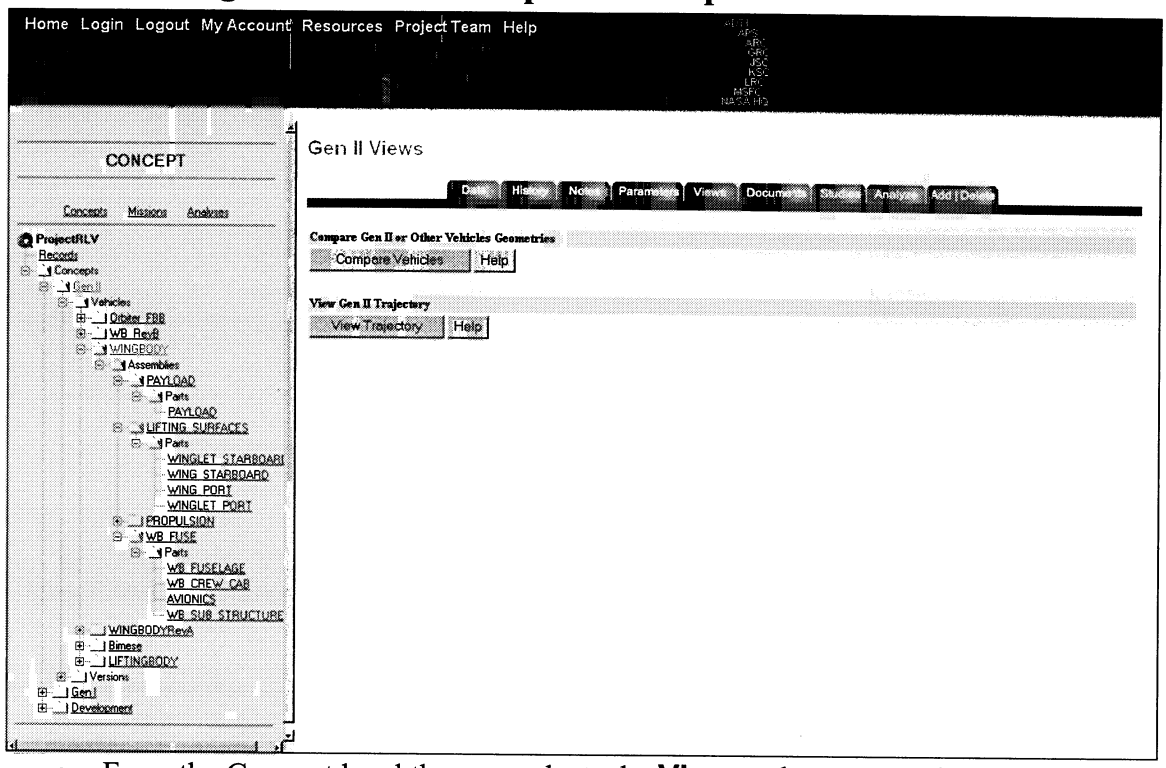
Name	Value	Units	Description	Type	Code Variable
Quantity	1	-	Number of entities	CAD	-
Isp_SL	418.	lbf-sec/lbm	Engine Isp Sea Level	HAVOC-in	RISPSL
Isp_vac	435.	lbf-sec/lbm	Engine Isp vacuum	HAVOC-in	RISPVA
altitude_orbit	1336700.	ft	Orbit altitude	HAVOC-in	HF
inclination_orbit	51.6	deg	Orbit inclination	HAVOC-in	CHIORB
mass_payload	148415.	lbm	Propulsion weight	HAVOC-ou	WPS
empty_weight_booster	-	lbm	Empty booster weight	HAVOC-ou	-
empty_weight_airframe	476781.	lbm	Empty airframe weight	HAVOC-ou	-
weight_airframe	244811.	lbm	Airframe Weight	HAVOC-ou	WAF
weight_propulsion	156710.	lbm	Propulsion weight	HAVOC-ou	-
weight_engine	-	-	Engine weight	HAVOC-ou	-
weight_fixed_equipment	38381.	lbm	Fixed Equipment Weight	HAVOC-ou	WFIXEQP
WE/GLOW	0.081	-	WE/GLOW	HAVOC-ou	WEGLOW
length_payload	60	ft	Payload length	HAVOC-in	PAYLOAD
diameter_payload	20	ft	Payload diameter	CAD	-

3.11 Parameters Lower Portion of Window



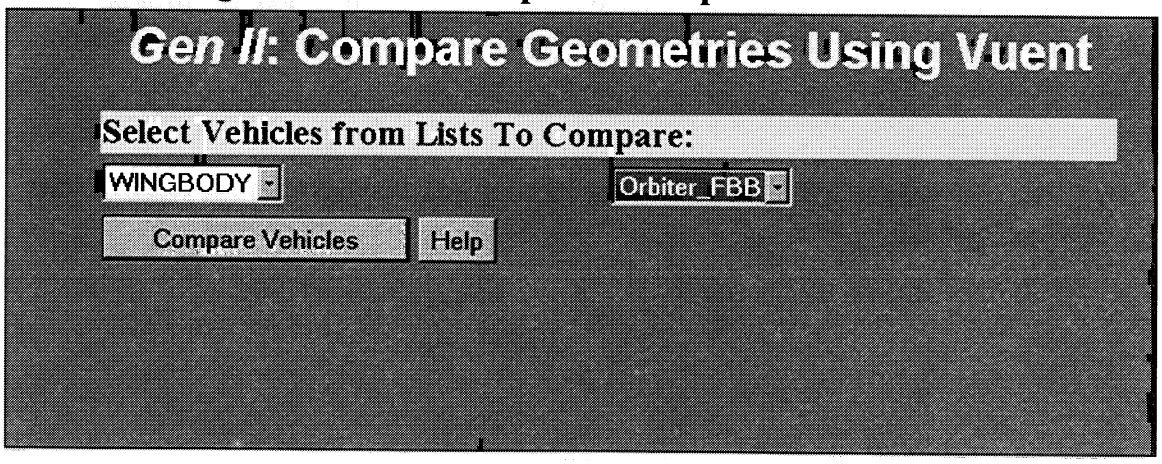
- The Parameters tab can be used to initiate a change in a geometry or an analysis variable.
- The user is required to create a new vehicle if a geometric parameter is modified.
- After making the desired changes the user clicks the **Modify** button to update the database.
- If a geometric change has been made the **Modify** button will automatically activate the CAD program (ProE) which will create the new geometry based on the user's inputs.
- With a geometry update the geometric files required to run the analysis codes (HAVOC is currently the only one available) will be created.
- The new vehicle will be available for viewing and analysis after the database has been updated.

3.12 Selecting Vehicles for Comparison Step One



- From the Concept level the user selects the **Views** tab to access the vehicle comparison function.
- Only the **Compare Vehicles** button is operational on this page.

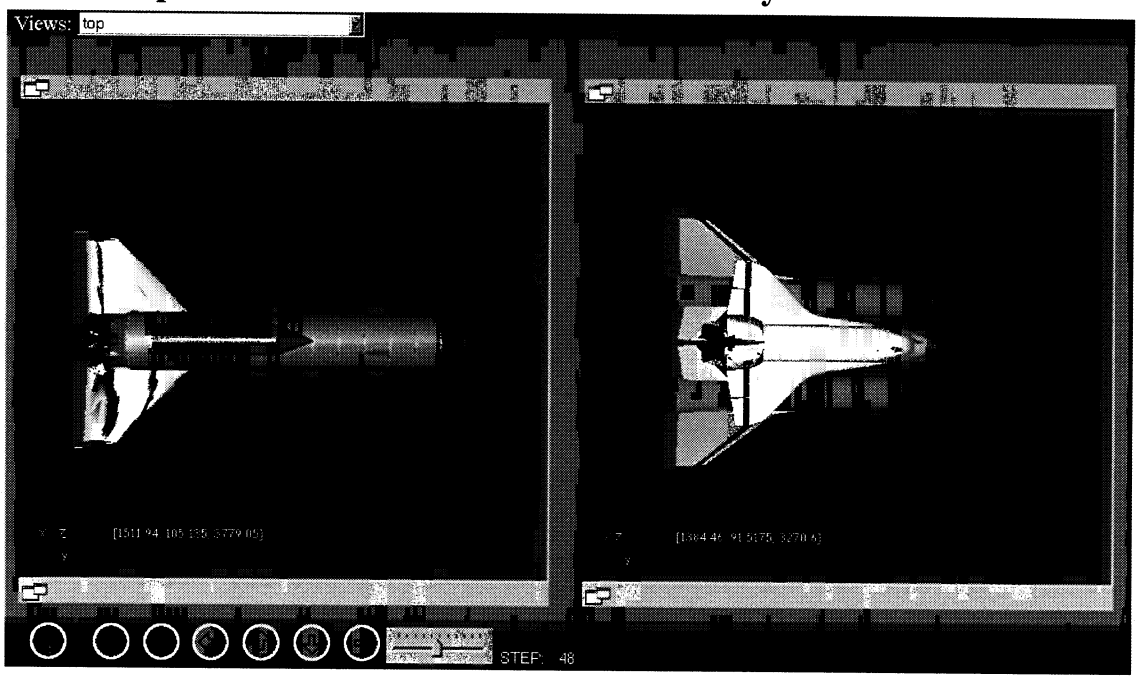
3.13 Selecting Vehicles for Comparison Step Two



- One vehicle from each list is selected and then the user clicks the **Compare Vehicles** button to bring up a new window with the comparison views.

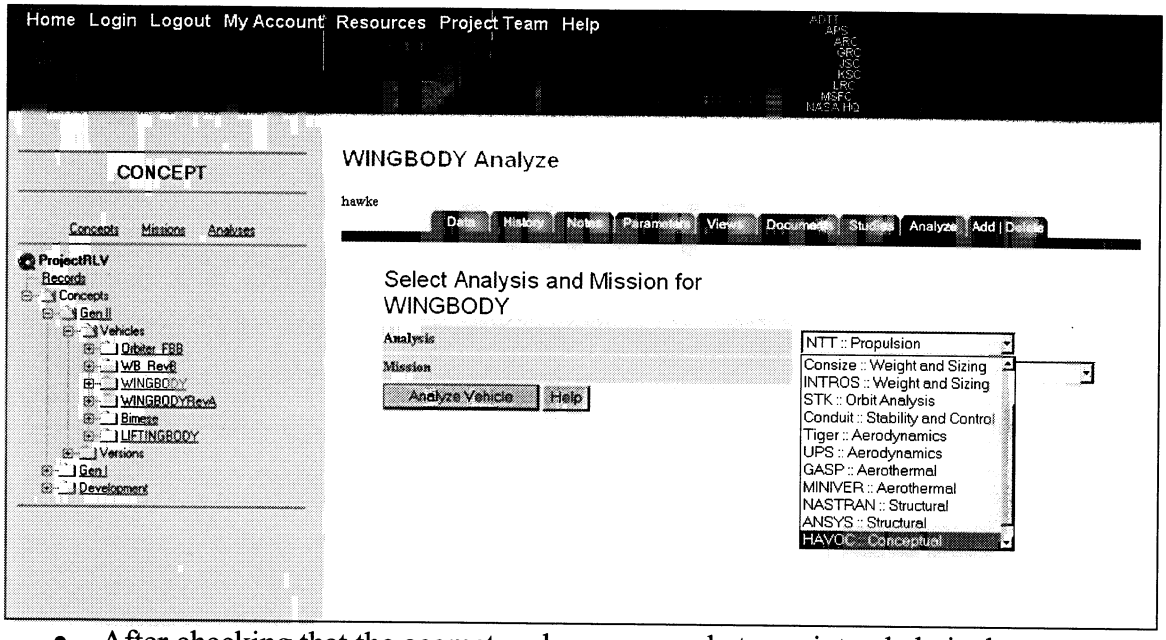
- The newly created geometry with the computer or user generated name would be available at this point for comparison.

3.14 Comparison of New and Baseline Geometry



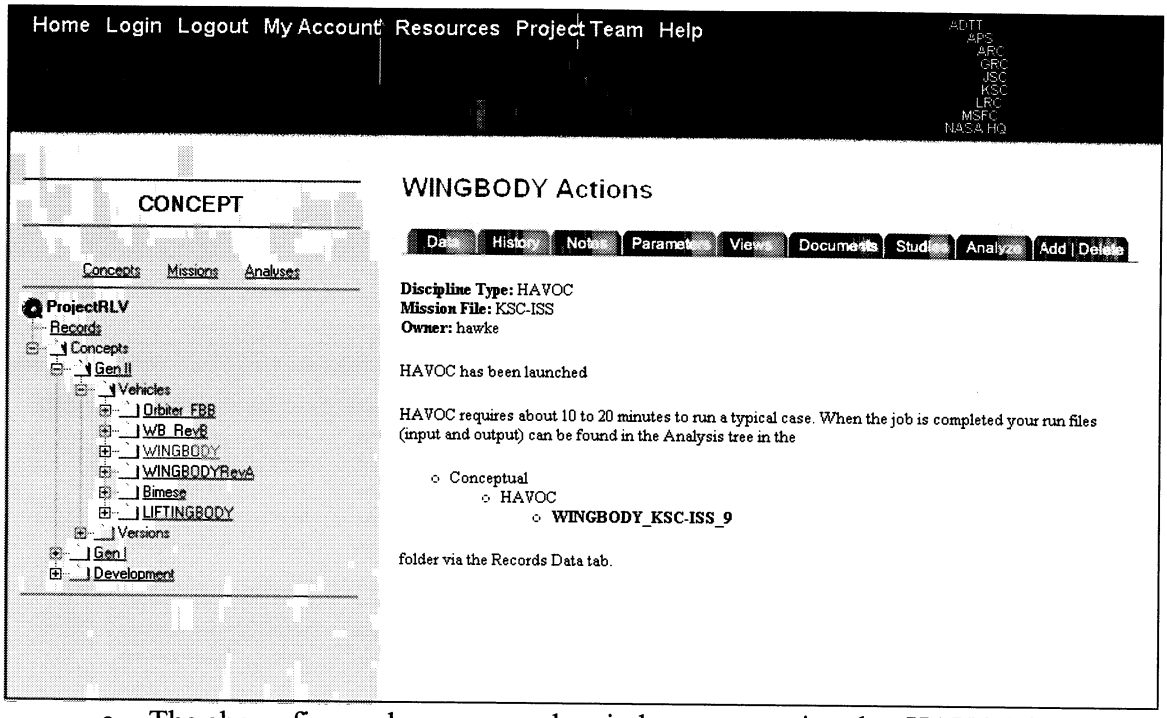
- The user can select from the list of views which will appear in both windows.
- The lower left hand buttons generally operate on both geometries, though there are some bugs to be worked out in this function.
- The views can be manipulated independently within the comparison window or detached from the parent window via the  button.

3.15 Starting an Analysis Run



- After checking that the geometry change was what was intended via the comparison view or standard view of the geometry the next step may be to make an analysis run.
- The above sample shows the selection of **HAVOC** for the analysis and **KSC-ISS::Ascent** for the mission.
- After clicking the **Analyze Vehicle** button there is a waiting period while the database is updated and the analysis is started.

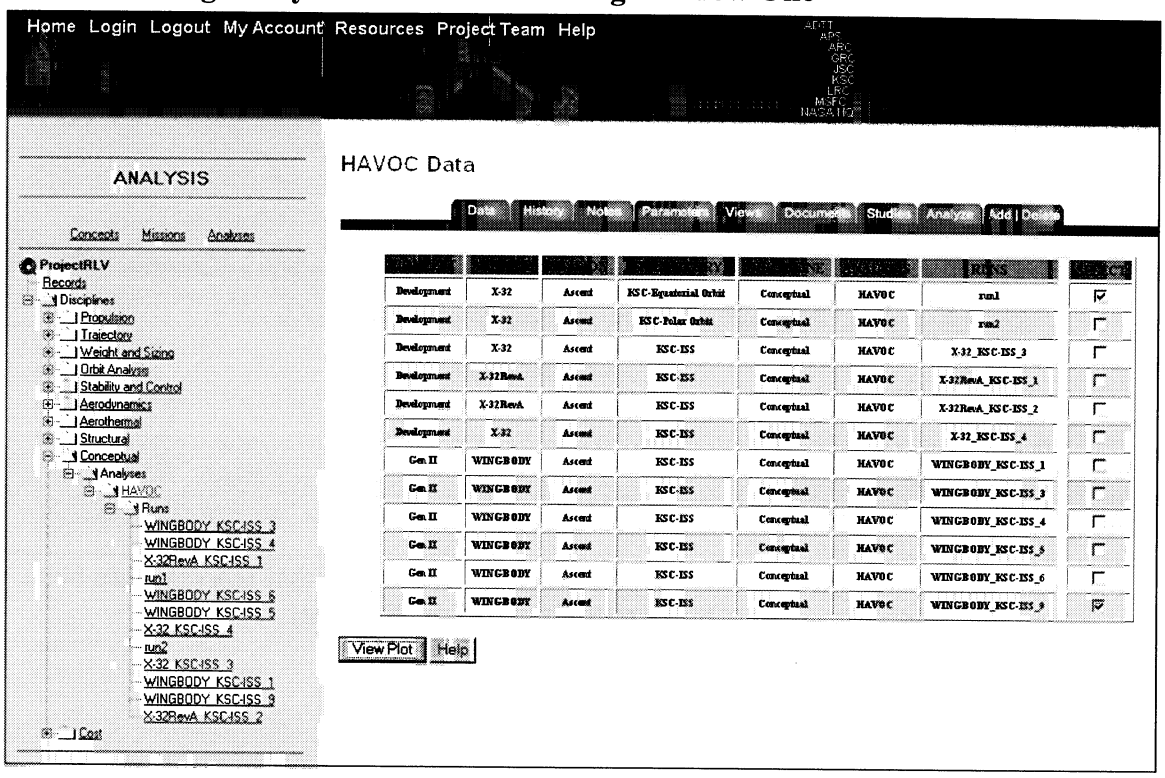
3.16 Analysis Launched Window



- The above figure shows a sample window announcing that HAVOC has been launched for a given vehicle. (WINGBODY in this case)

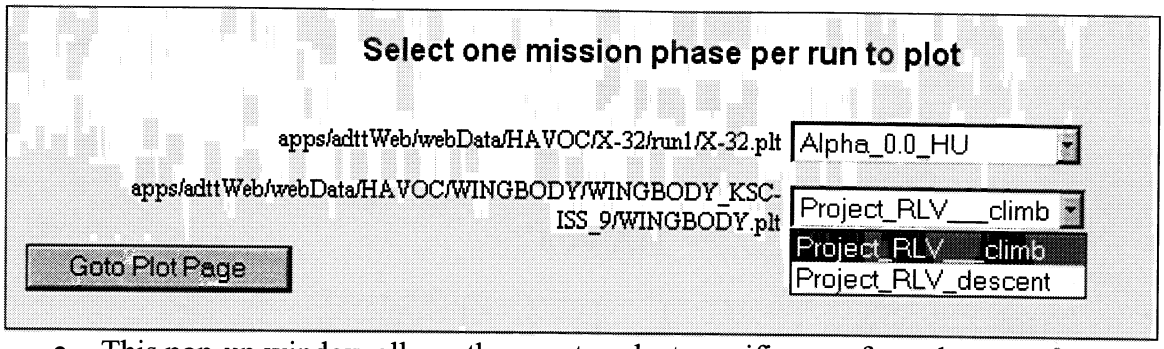
3.17 2D Plotting Windows

3.17.1 Viewing Analysis Data via 2D Plotting Window One



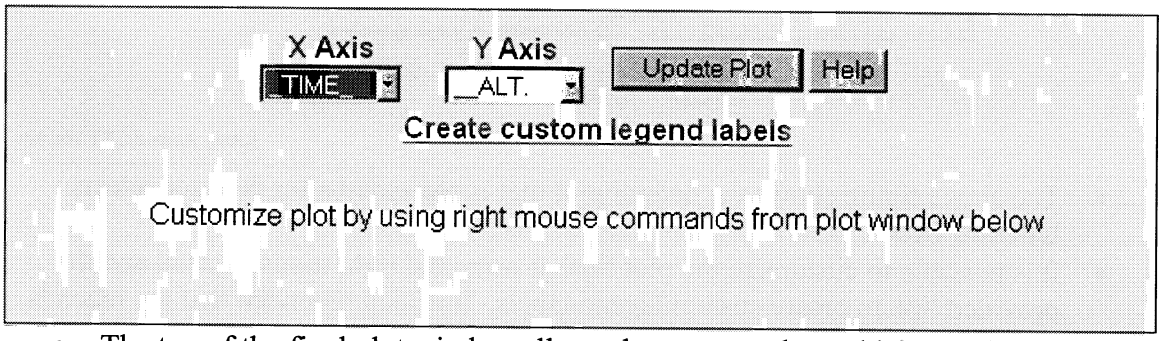
- Once the analysis run is complete the data will be available in a couple of locations.
- To compare the results to other cases of the same code that were run previously, the user can get to the above window with the following steps:
  - Select **Analyses** from the **CONCEPTS** page. This will pop up a new window labeled **ANALYSIS**.
  - Navigate through the tree to the folder showing the desired analysis.
  - Select the **Data** tab.
  - Select the desired runs in the last column under **SELECT**.
  - Click on the **View Plot** button.

3.17.2 Viewing 2D Plotting Window Two



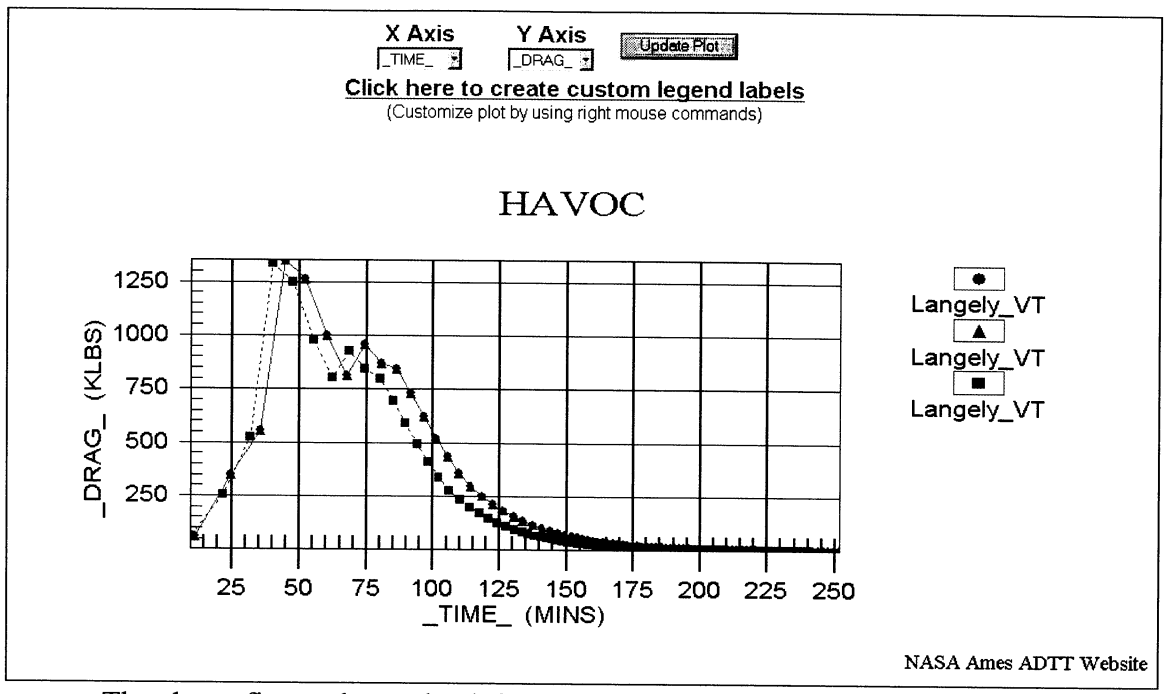
- This pop-up window allows the user to select specific runs from the sets of runs available within a HAVOC analysis.
- After making the selection the user clicks on **Goto Plot Page** to continue to the next window.

3.17.3 Top of 2D Plot Window Three



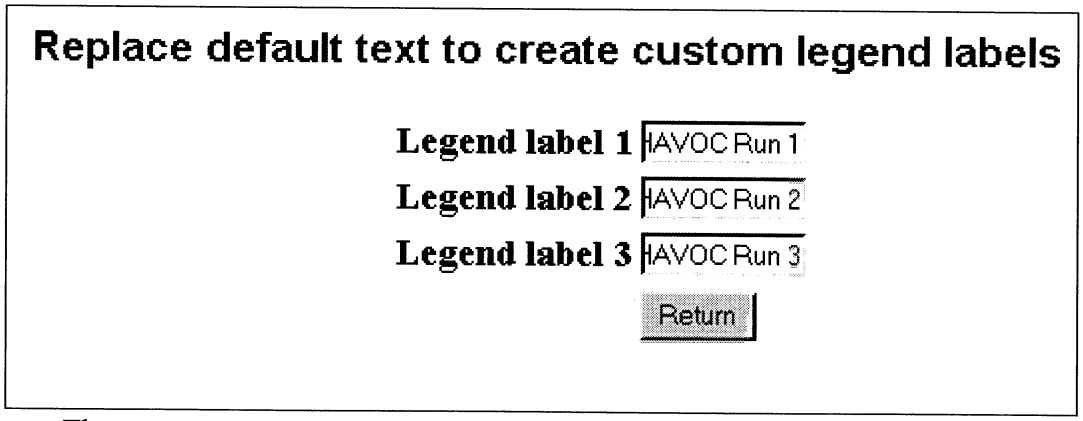
- The top of the final plot window allows the user to select which variables are to be plotted from a list of available parameters.
- The plot is filled in the lower portion of the window after selecting **Update Plot**.

3.17.4 Sample 2D Plot Initial Format



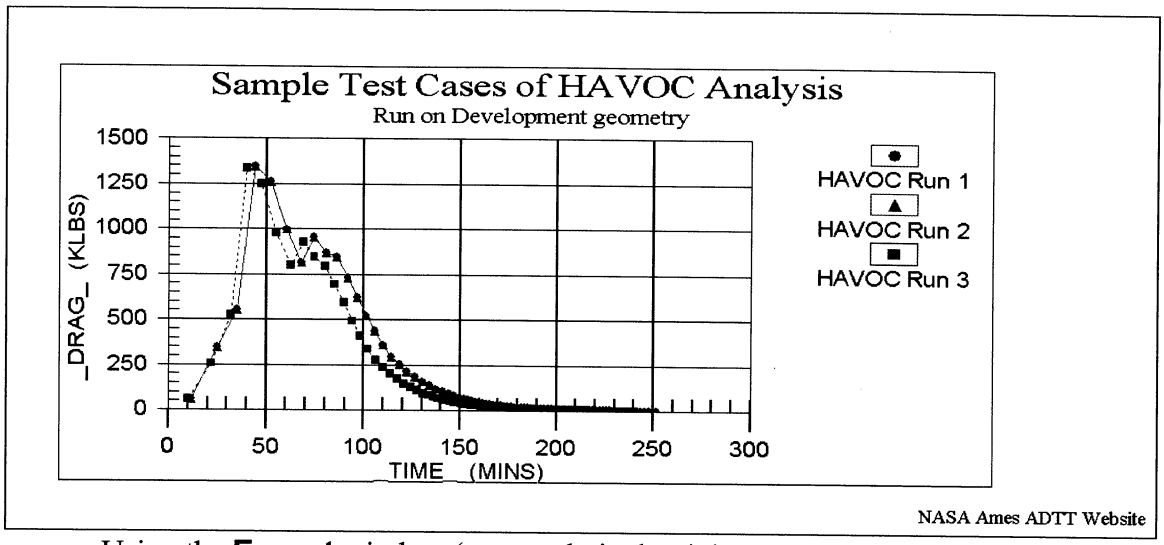
- The above figure shows the default plot that is presented after the user selects **Update Plot**
- It is possible to customize the plot further using the right mouse button and the provided link to create custom legend labels. It is recommended that the legends be customized first as a legend change requires the plot to be updated again to take effect, causing any changes on the main plot made to that point to be reset.

3.17.5 Legend Update Window



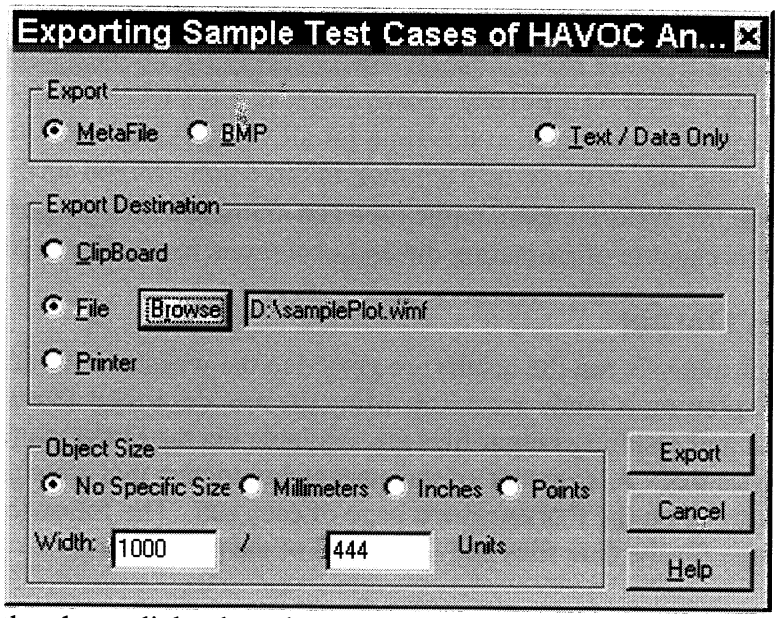
- The user can enter up to 30 characters per legend using the **Return** button to get back to the plotting page.
- The user will need to click the **Update Plot** button for the changes to the legend to take effect.

3.17.6 Sample Final Plot



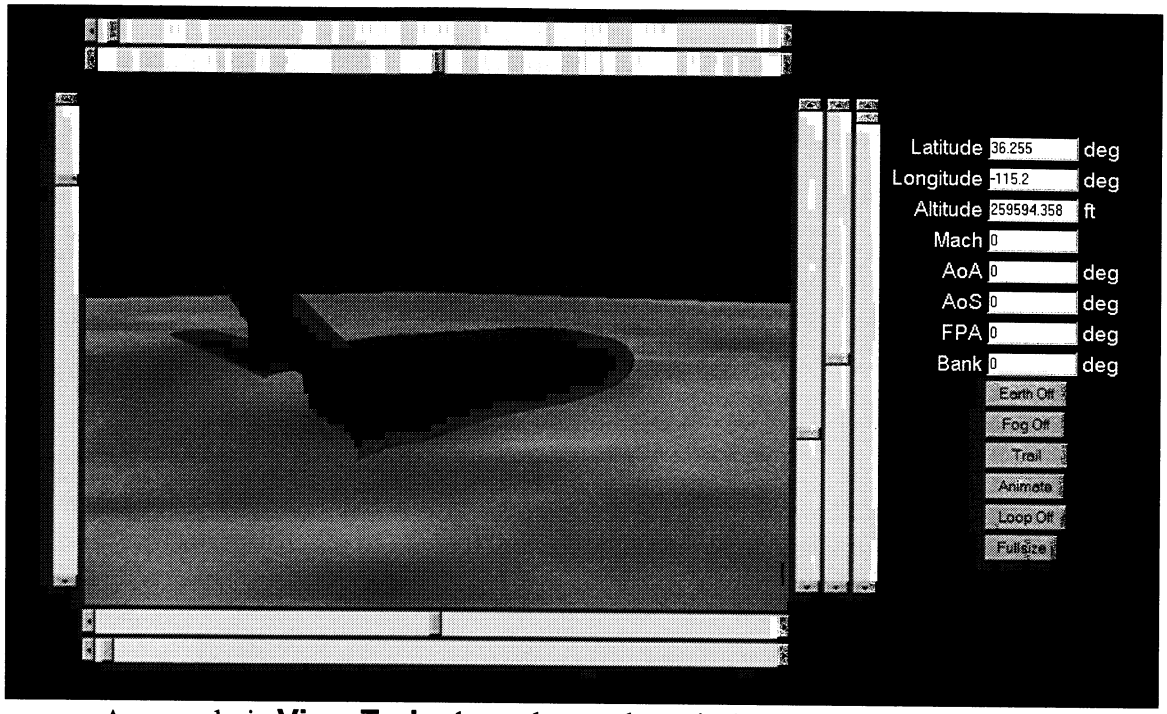
- Using the **Export** window (accessed via the right mouse button) shown below the user has a few options on saving the plot and its data to other formats.

3.17.7 Plot 2D Export Window



- As seen in the above dialog box the user has the choice of a few different export formats.

3.18 3D Solution Viewing

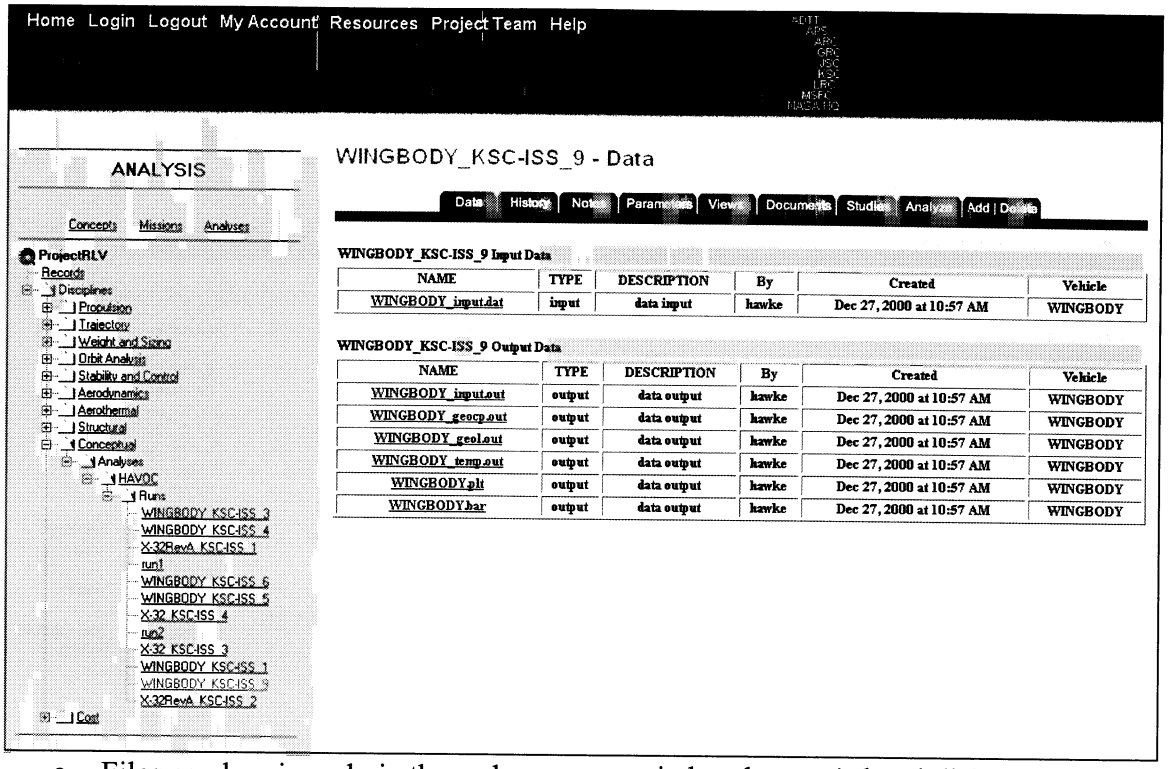


- Accessed via **View Trajectory** shown above in 3.6 Views Window.
- Though not fully functional at this time for all vehicles and solutions, this tool will allow the user to examine a solution in three dimensional form.
- In addition the Wild Tangent application can animate a mission path.
- This tool is capable of animating different solutions for each step of a mission.

3.19 Analysis Output

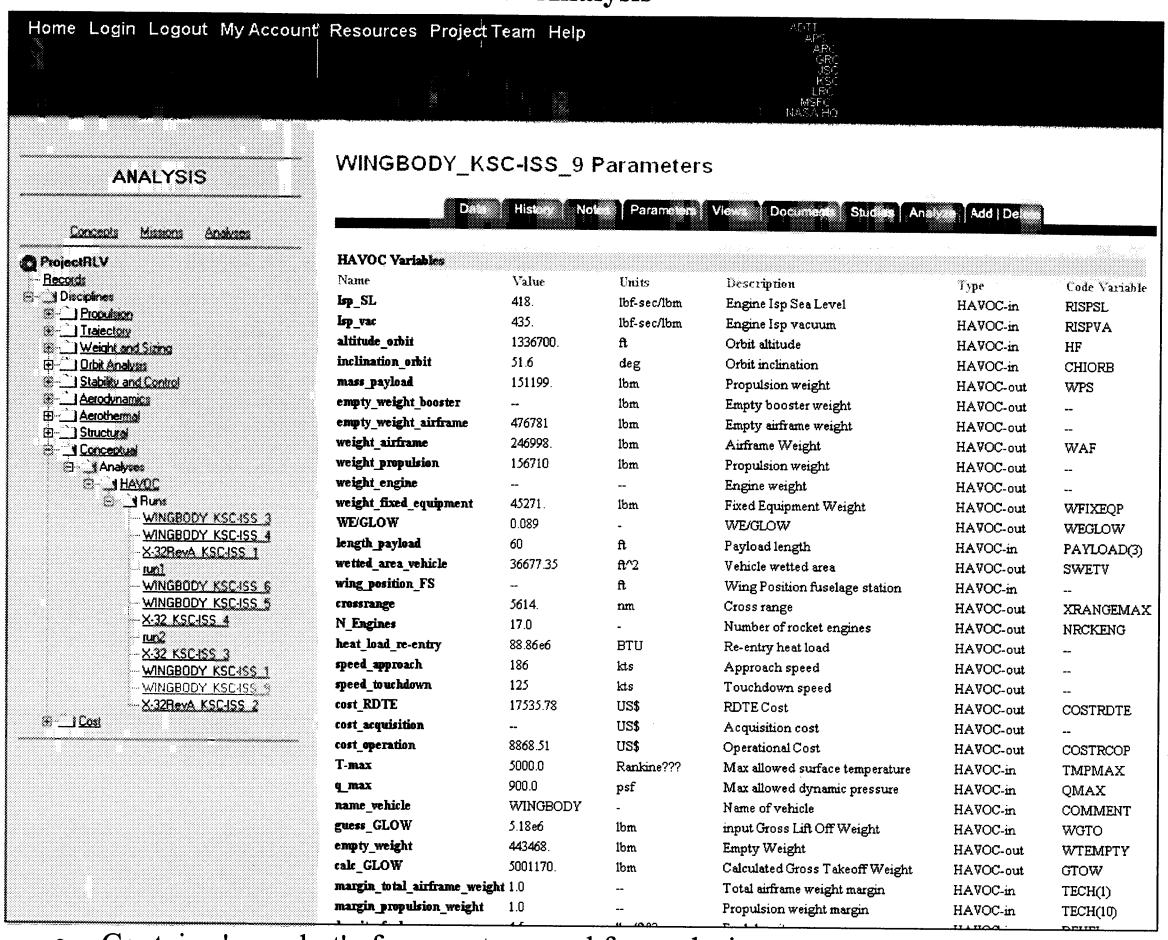
After each analysis run the results are posted to the website in three places. The data used for 2D plotting is made accessible using the tool described above. In addition the original input and resulting output files are available via the **Data** tab for each run within the Analysis tree. A 'snapshot' of the current parameters that were used for the analysis are preserved on the **Parameters** page of each run. The following two figures show the web page for the data files and the record of the parameters.

3.19.1 Analysis Input/Output files



- Files can be viewed via the webpage or copied to the user's local disk.

3.19.2 Record of Parameters Used for Analysis



Name	Value	Units	Description	Type	Code Variable
Isp_SL	418.	lbf-sec/lbm	Engine Isp Sea Level	HAVOC-in	RISPSL
Isp_vac	435.	lbf-sec/lbm	Engine Isp vacuum	HAVOC-in	RISPVA
altitude_orbit	1336700.	ft	Orbit altitude	HAVOC-in	HF
inclination_orbit	51.6	deg	Orbit inclination	HAVOC-in	CHIORB
mass_payload	151199.	lbm	Propulsion weight	HAVOC-out	WPS
empty_weight_booster	--	lbm	Empty booster weight	HAVOC-out	--
empty_weight_airframe	476781	lbm	Empty airframe weight	HAVOC-out	--
weight_airframe	246998.	lbm	Airframe Weight	HAVOC-out	WAF
weight_propulsion	156710	lbm	Propulsion weight	HAVOC-out	--
weight_engine	--	--	Engine weight	HAVOC-out	--
weight_fixed_equipment	45271.	lbm	Fixed Equipment Weight	HAVOC-out	WFIXEQP
WE/GLOW	0.089	-	WE/GLOW	HAVOC-out	WEGLOW
length_payload	60	ft	Payload length	HAVOC-in	PAYLOAD(3)
wetted_area_vehicle	36677.35	ft^2	Vehicle wetted area	HAVOC-out	SWETV
wing_position_FS	--	ft	Wing Position fuselage station	HAVOC-in	--
crossrange	5614.	nm	Cross range	HAVOC-out	XRANGEMAX
N_Engines	17.0	-	Number of rocket engines	HAVOC-out	NRCKENG
heat_load_re-entry	88.86e6	BTU	Re-entry heat load	HAVOC-out	--
speed_approach	186	kts	Approach speed	HAVOC-out	--
speed_touchdown	125	kts	Touchdown speed	HAVOC-out	--
cost_RDTE	17535.78	US$	RDTE Cost	HAVOC-out	COSTRDTE
cost_acquisition	--	US$	Acquisition cost	HAVOC-out	--
cost_operation	8868.51	US$	Operational Cost	HAVOC-out	COSTRCOP
T-max	5000.0	Rankine???	Max allowed surface temperature	HAVOC-in	TMPMAX
q_max	900.0	psf	Max allowed dynamic pressure	HAVOC-in	QMAX
name_vehicle	WINGBODY	-	Name of vehicle	HAVOC-in	COMMENT
guess_GLOW	5.18e6	lbm	input Gross Lift Off Weight	HAVOC-in	WGTO
empty_weight	443468.	lbm	Empty Weight	HAVOC-out	WTEMPTY
calc_GLOW	5001170.	lbm	Calculated Gross Takeoff Weight	HAVOC-out	GTOW
margin_total_airframe_weight	1.0	--	Total airframe weight margin	HAVOC-in	TECH(1)
margin_propulsion_weight	1.0	--	Propulsion weight margin	HAVOC-in	TECH(10)

- Contains 'snapshot' of parameters used for analysis.
- Only includes input and output parameters for the analysis. Geometric parameters for the vehicle can be found in the Concepts tree under the given vehicle's Parameters tab.

4.0 Tools

All of the tools connected to the ADTT Website are accessible thorough the Internet. Some of the packages require a client application be downloaded to allow the local computer to interact properly with the web based application. There are five general categories of tools used for this project:

1. geometry generation
2. geometry inspection
3. solution and geometry comparison
4. collaboration
5. data organization and management

In the following subsections, the tools of each category that are used in this build of ADTT are described. Recommendations for future development are provided. A summary of the third party tools both considered and incorporated can be found in Appendix A.

4.1 Geometry Generation

In order to provide the framework for an iterative design process the capability to update the underlying CAD geometry is needed. This requires that the geometry be regenerated every time a user selects a geometric change. These changes are recorded in the database as new vehicles. The new CAD geometry is used as input to the appropriate analyses and it provides updated information to the geometry viewing tools.

Two widely used CAD packages, ProEngineer and Catia, provided the basis for the initial demonstration. While these were the only two initially incorporated, consideration was given to other CAD packages so that it would be possible to include any CAD package that is commonly used for design. This section describes some of the details of the CAD packages evaluated with respect to their function in the overall design process.

4.1.1 ProEngineer

Pro Engineer was used to build most of the geometry for 4 candidate 2nd generation RLV concepts. ProProgram can be used to facilitate regeneration using new parameter values read from an input file. A trail file is recorded for such an update and for all geometry outputs required by analysis programs (such as triangulation in the Nastran format, IGES or Inventor files). New values for geometric parameters are read from the database and written into the appropriate input file (using Java or Perl utilities for the data format translation). Limited customization is still required for each concept that is being studied, but the ability to update geometry through the ADTT environment for pre-existing ProEngineer geometry can be achieved within a couple of days. Full geometry updating has been demonstrated for the Wing Body geometry. ProEngineer running on a Windows NT system was invoked from a Unix system, and files were transferred in both directions.

Automated export of geometry points files for HAVOC input was demonstrated

for the Wing Body geometry. Planar cuts are made on the body and wings, and appropriate numbers of points are distributed around each cut (50 points for the body, 40 points for the wing). An IGES points file is exported. A Perl script for reading IGES points and reordering for HAVOC (body points starting at the starboard maximum width and proceeding up and around the body, wing points starting at the leading edge) has been completed. Slight customization is required to change input and output file names and locations, but the core conversion algorithm is generic. Details on the methods used for automation of the ProE geometry to HAVOC input conversion can be found in Reference 3.

Some work was completed on a generic part library for space launch vehicles. Generic fuselages, wings and rocket nozzles were constructed. Each can be modified parametrically. Assembly of components into a vehicle was not automated: CAD specialists are still required to select and locate the components, and a custom interface to drive the relevant parameters must be completed for each vehicle concept.

4.1.2 CATIA

For the demonstration a small number of geometric parts were created in CATIA to show compatibility with the 3D visualization tool as well as to show the ability to incorporate a variety of CAD packages. It was shown that a geometry modification can be scripted and therefore automated through the web parameter modification form. More details about this process can be found in Reference 4. As this capability required more extensive programming than was practical for the initial demonstration it was not fully incorporated into the current version of the website.

4.1.3 CAD Packages Recommendations

ProEngineer is well-suited to parametric update. Catia can be driven in this fashion for some geometry changes, but it is unwieldy. SDRC-IDEAS and Unigraphics CAD systems can be included as appropriate. Some facility for dealing with IGES and STEP data from external sources should be added.

The component library should be expanded to have functionality similar to the RAM geometry system. It is difficult to create generic parts in ProEngineer using the same parameterization as RAM, but the potential overlap should be fully explored.

4.2 Inspection

Another component of design is the ability to inspect the results of an updated vehicle. The users need access to tools that allow them to view the new geometry as well as the results of any analysis that they run to evaluate the proposed modification.

4.2.1 3D Geometry Viewer

The following sections describe the current 3D geometry viewing tool as well as other applications that were evaluated in the course of putting together the initial demonstration. Basic requirements for the tool include the ability to read or translate

from standard native CAD packages and a way for the primary viewer to run a viewing session through the website. Other considerations include ease of use and degree of functionality.

4.2.1.1 Vuent/iEngineering/EnvisionI

Vuent software was selected for initial integration as the website 3D viewing tool as well as collaboration/annotation facilitator. A detailed evaluation of this package is given in References 1 and 2. The Vuent application includes CAD export/translation features and provides a collaborative environment that allows saving annotations for future reference. Though it has the basic items of interest for the advance design tool, it's implementation and interface are more difficult to use than most of the other packages. If this tool is selected for the production version of the ADTT Website, it is recommended that some effort be made to encourage Vuent(iEngineering) to provide a more user friendly environment.

4.2.1.2 Other Possibilities

A wide variety of other 3D viewing tools are available, though few bring together all of the aspects required for the ADTT Website. A summary of the other tools evaluated can be found in spreadsheet form in Appendix A. One possible alternative to the Web based 3D viewing capabilities of Vuent may soon be available in Actify's 3D View application. In the evaluation, this tool stood out in ease of use and in providing the user with a good handful of tools useful for evaluating and annotating geometry as well as being able to read native CAD files directly.

4.2.2 3D Geometry Viewer Recommendations

New web based CAD independent geometry viewing tools are becoming available on a regular basis. It is likely that one of these tools will provide a better platform for sharing and viewing model geometries than Vuent. It is recommended that some effort be spent in continuing to evaluate the effectiveness and functionality of these tools so that the right software can be incorporated in to the working version of the website.

4.3 Comparison

In order to evaluate and analyze the merits of each new proposed change a user should be able to compare the baseline and updated geometries side-by-side. This requirement is needed for comparing the geometric changes as well as the changes found through the running of various analysis codes.

4.3.1 Geometric Comparisons via Vuent

As an add on to the 3D viewing tool described above in section 4.2.1.1 the capability to compare 3D geometries side by side was included. The comparison page is accessed at the concept level and allows to user to select up to two vehicles to compare. A separate window containing two side-by-side Vuent sessions allows the user to

manipulate the views with a pull down menu. The navigation tools provided work only marginally, with some problems synchronizing the two inner viewing windows.

4.3.2 Analysis Codes

It is the eventual intent to allow a user to launch a wide variety of analysis codes via a webpage form. To show the proof of concept the conceptual analysis code HAVOC was chosen to demonstrate how a code can be launched on a remote machine based on a users input through the website. The results are then posted back to the website and can be viewed in their raw form or through the 2D and 3D plotting tools described below.

4.3.2.1 HAVOC

HAVOC is a conceptual design code. The input is based on data stored for a given geometry. HAVOC input parameters can be modified by the users to do sensitivity studies. New sets of the geometry input for HAVOC are created and stored each time a vehicle is geometrically modified (resulting in a new vehicle). This code has been integrated to the ADTT Website providing users with the capability of modifying and adding parameters for input, launching a run and then of viewing the resulting output in both graphical and text forms.

4.3.3 Analysis Codes Recommendations

Placeholders have been provided for setting up links to a variety of design codes. The upcoming Son of HAVOC project should be able to be fully incorporated in to a web based design.

4.3.4 2D Plotting

The intent for this tool was to provide the user with the capability of plotting data from the analysis outputs.

4.3.4.1 ProEssentials

ProEssentials by GigaSoft has been incorporated in to an interactive webpage to provide the user with 2D plotting capabilities. It is run using Visual Basic scripting created through a parent JAVA script on the fly as the user requests different files and datasets. The web based application allows for interactive changes such as rescaling and symbol choice for the datasets. An export dialog allows the user to download the data to a standard meta file, text file or bitmap picture file. The user can compare time histories and bar charts that show sensitivities based on both absolute and percentage change of selected parameters.

4.3.5 2D Plotting Recommendations

The software package evaluated and incorporated provides a reasonable level of scientific plotting capability. It is possible that other 2D plotting languages could be incorporated if needed in the future, however at this time the current package is considered to be satisfactory. Some of the elements that are missing which may be

desired in the future include things like minor grids and a wider range of symbols to choose from, and standard formatting templates.

4.3.6 3D Solution Viewing

Although the variety of applications for 3D solution viewing is extensive, the technology to view analytical solutions on a 3 dimensional geometry via a webpage is only in its developmental stage. The tool currently incorporated is described below.

4.3.6.1 Wild Tangent

This software provides solution viewing mapped to 3D geometry. The code is written in JAVA script and allows for versatility in the interface and information plotted. WildTangent's Web Driver is a 480K, Internet browser plug-in that piggybacks onto Internet Explorer or Netscape Navigator. The Web Driver provides additional functionality that allows 3D graphics (games, music visualizers and mapping applications) to play on a computer's multimedia hardware (video and sound card, CPU, memory) while on the Internet.

4.3.7 3D Solution Recommendations

Wild Tangent already provides a good tool for viewing solutions on 3D geometry. With some development it could show various solutions along a trajectory. Some research could be done to explore other possibilities, though at this time WildTangent is a good fit for this functionality.

4.4 Collaboration and Annotation

One of the primary requirements for the ADTT Website is to provide a collaborative environment for the design team. A tool that allows the team members to work together in real time across the various sites is an integral part of the functionality of the website. In addition the collaboration and annotation tools should provide a means for the users to save and document their sessions for future reference.

4.4.1 Vuent

The collaborative session in Vuent is provided through the 3D viewing tool, allowing for a geometry based communication session. Details on the demonstrated functionality and evaluation of this tool are in Reference 1. Similar comments to those given in section 4.2.1, regarding the 3D geometry viewing tool, apply here as well.

4.4.2 Other Possibilities

Other applications were evaluated including websites such as Groove and Cahoots. These allow for group discussions as well as project document posting. They also provide a means for users to drive a guided tour of webpages, allowing a primary user to control the screens of the other users.

Microsoft NetMeeting facilitates connections directly between IP addresses and has a handful of useful features such as a whiteboard that supports pasted images from

other programs, chat windows and file transfers. The chat sessions and whiteboard can be saved for future reference.

Other geometry viewing packages advertise the ability to collaborate between users. Actify's Spin application has provisions for marking up the geometry with pointers and text, but not yet freehand drawing.

4.4.3 Collaboration Recommendations

Further evaluation of the available products for this feature is recommended. It is likely that an application with a smoother and more useful interface than Vuent's can be found. Both stand alone collaborative tools as well as 3D geometry viewing tools with built in collaboration/annotation features should be considered.

4.5 Database

The underlying structure and organization of the website is provided using a database.

4.5.1 ODBMS ObjectStore

The *ManagerObjectStore* interface is used to open or create a database. Currently, the interface only allows a single session to the database.

com.nasa.database.ostore
Class ManagerObjectStore

```
java.lang.Object
  |
  +--com.nasa.database.ManagerDatabase
        |
        +--com.nasa.database.ostore.ManagerObjectStore
```

ManagerObjectStore interface allows users to access to the object database for UPDATE or READONLY via methods:

- ***persist****(com.nasa.attribute.Attribute pAttribute)* saves an object to database
- ***retrieveList****(com.nasa.attribute.Attribute pAttribute)* return vector list of object that satisfies the input pAttribute.
- ***delete****(com.nasa.attribute.Attribute pAttribute)* removes an object from the database.

Note that the *retrieveList()* method consumes a substantial amount of time due to an insufficient algorithm.

4.5.1.1 Structural Data:

The basic structural data for the project has been mapped into an object database via an existing API. The RLVCollection interface is a centralized object that allows all of the structural objects (e.g. Project, Concept, Vehicle, Subassembly, Part, Discipline, Analysis, Run, Mission, Trajectory, and TrajectoryPoint) to have data attributes (e.g. data, parameter, notes, history, documents, geometries, etc.)

com.nasa.rlv
Class RLVCollection

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
```

Direct Known Subclasses:

Analysis, Concept, Discipline, Mission, NameValue, Part, Project, Run, Subassembly, Trajectory, TrajectoryPoint, TrajectoryPointValue, Vehicle, Version

All of the objects that extended from RLVCollection can access their data attributes via getter/setter methods.

1. Project:

com.nasa.rlv
Class Project

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Project
```

To retrieve all of the project objects in the database, one can use:

- ***ManagerObjectStore.retrieveList****(new Project())* returns a vector list of all project.

The Project object is the highest hierarchical order in the structural data. It has Concept, Discipline, and Mission as its direct children, as mentioned in the database schema. To add a child to the Project object, one can use:

- ***addConcept****(Concept pConcept)*
- ***addDiscipline****(Discipline pDiscipline)*
- ***addMission****(Mission pMission)*

To query a child of the project, one can use:

- ***getConceptItem****(int pIndex)* returns a Concept object
- ***getDisciplineItem****(int pIndex)* returns a Discipline object
- ***getMissionItem****(int pIndex)* returns a Mission object

2. Discipline:

com.nasa.rlv

Class Discipline

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Discipline
```

To retrieve all of the discipline objects in the database, one can use:

- ***ManagerObjectStore.retrieveList****(new Discipline())* returns a vector list of all discipline objects.

Discipline objects are under a project object. To query or add *Discipline* objects that belong to a project, one can use:

- ***getDisciplineItem****(int pIndex)* or ***getDisciplineItem****(java.lang.String pValue)* returns discipline object.
- ***getDisciplineSize()*** returns number of discipline objects of the project.

3. Analysis:

com.nasa.rlv

Class Analysis

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Analysis
```

To retrieve all of the *Analysis* objects in the database, one can use:

- ***ManagerObjectStore.retrieveList****(new Analysis ())* returns a vector list of all *Analysis* objects.

Analysis objects are under a *Discipline* object. To query or add *Analysis* objects that belong to a discipline, one can use:

- ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns a RLVCollection object that can be cast to *Analysis* object.
- ***getCollectionSize()*** returns number of *Analysis* objects of the discipline.

4. Run:

com.nasa.rlv
Class Run

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Run
```

To retrieve all of the *Run* objects in the database, one can use:

- ***ManagerObjectStore.retrieveList****(new Run ())* returns a vector list of all of the *Run* objects.

Run objects are under the *Analysis* object. To query or add *Run* objects that belong to a analysis, one can use:

- ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns a RLVCollection object that can be cast to *Run* object.
- ***getCollectionSize()*** returns number of *Run* objects of the analysis.

5. Mission:

com.nasa.rlv
Class Mission

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Mission
```

To retrieve all of the mission objects in the database, one can use:

- ***ManagerObjectStore.retrieveList****(new Mission())* returns a vector list of all mission objects.

Mission objects are under a project object. To query or add *Mission* objects that belong to a project, one can use:

- ***getMissionItem****(int pIndex)* or ***getMissionItem****(java.lang.String pValue)* returns *Mission* object.

- § ***getMissionSize()*** returns number of *Mission* objects of the project.

6. Trajectory:

com.nasa.rlv

Class Trajectory

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Trajectory
```

To retrieve all of the *Trajectory* objects in the database, one can use:

- § ***ManagerObjectStore.retrieveList****(new Trajectory ())* returns a vector list of all *Trajectory* objects.

Trajectory objects are under a *Mission* object. To query or add *Trajectory* objects that belong to a mission, one can use:

- § ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns a RLVCollection object that can be cast to *Trajectory* object.
- § ***getCollectionSize()*** returns number of *Trajectory* objects of the mission.

7. TrajectoryPoint:

com.nasa.rlv

Class TrajectoryPoint

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.TrajectoryPoint
```

To retrieve all of the *TrajectoryPoint* objects in the database, one can use:

- § ***ManagerObjectStore.retrieveList****(new TrajectoryPoint ())* returns a vector list of all *TrajectoryPoint* objects.

TrajectoryPoint objects are under a *Trajectory* object. To query or add *TrajectoryPoint* objects that belong to a trajectory, one can use:

- § ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns a RLVCollection object that can be cast to *TrajectoryPoint* object.

§ ***getCollectionSize()*** returns number of *TrajectoryPoint* objects of the trajectory.

8. Concept:

com.nasa.rlv

Class Concept

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Concept
```

To retrieve all of the concept objects in the database, one can use:

§ ***ManagerObjectStore.retrieveList****(new Concept())* returns a vector list of all concept objects.

Concept objects are under a project object. To query or add *Concept* objects that belong to a project, one can use:

§ ***getConceptItem****(int pIndex)* or ***getConceptItem****(java.lang.String pValue)* returns Concept object.

§ ***getConceptSize()*** returns number of concept objects of the project.

9. Vehicle:

com.nasa.rlv

Class Vehicle

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Vehicle
```

To retrieve all of the *Vehicle* objects in the database, one can use:

§ ***ManagerObjectStore.retrieveList****(new Vehicle())* returns a vector list of all *Vehicle* objects.

Vehicle objects are under a *Concept* object. To query or add *Vehicle* objects that belong to a concept, one can use:

- § ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns an RLVCollection object that cast to *Vehicle* object.
- § ***getCollectionSize()*** returns number of Vehicle objects of the Concept.

10. Subassembly:

com.nasa.rlv

Class Subassembly

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Subassembly
```

To retrieve all of the *Subassembly* objects in the database, one can use:

- § ***ManagerObjectStore.retrieveList****(new Subassembly())* returns a vector list of all *Subassembly* objects.

Subassembly objects are under a *Vehicle* object. To query or add *Subassembly* objects that belong to a vehicle, one can use:

- § ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns an RLVCollection object that can be cast to *Subassembly* object.
- § ***getCollectionSize()*** returns number of Subassembly objects of the Vehicle.

11. Part:

com.nasa.rlv

Class Part

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVCollection
                          |
                          +--com.nasa.rlv.Part
```

To retrieve all of the *Part* objects in the database, one can use:

- § ***ManagerObjectStore.retrieveList****(new Part())* returns a vector list of all *Part* objects.

Part objects are under a *Subassembly* object. To query or add *Part* objects that belong to a subassembly, one can use:

- § ***getCollectionItem****(int pIndex)* or ***getCollectionItem****(java.lang.String pValue)* returns a RLVCollection object that can be cast to *Part* object.
- § ***getCollectionSize()*** returns number of *Part* objects of the subassembly.

4.5.1.2 Data Attributes:

1. *Parameter Attribute:*

com.nasa.rlv

Class RLVParameter

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVParameter
```

RLVParameter is the interface for parameter attributes where the parameter name serves as a key. The *RLVParameter* interface has setter/getter methods to access to its attributes, which are parameter name, parameter value, parameter units, parameter description, and parameter type (e.g. CAD, HAVOC-in, HAVOC-out), parameter changeable flag (e.g. yes or no), and parameter code variables which are variable names used in others legacy analysis codes.

To access to an object *parameter attribute*, one can use:

- § ***getParameter****(java.lang.String pName)* returns respective parameter (name, value, units, description, type, changeable flag, and code variable)
- § ***getParameterList()*** returns a vector of parameters
- § ***setParameterValue****(java.lang.String pName,*
  com.nasa.attribute.Attribute pValue,
  java.lang.String pUnit,
  java.lang.String pDesc,
  java.lang.String pType,
  java.lang.String pChange,
  java.lang.String pCode)
- § ***deleteParameter****(java.lang.String pName,*
  java.lang.String pUnit,
  java.lang.String pType) returns Boolean which indicates if the transaction is success.

2. Document Attribute:

com.nasa.rlv
Class RLVDocument

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVDocument
```

RLVDocument is the interface for document attributes where the document name serves as a key. The *RLVDocument* interface has setter/getter methods to access to its attributes, which are document name (name is made up of object name, random index, and type of document), document type (e.g. doc, gif, jpg, txt, dat, pdf, and other file type extensions), document description, document owner where only owner can remove the document from the database, document created date, and date of last access.

To access to an object *document attribute,* one can use:

- ***setDocumentValue****(java.lang.String pName,*
  java.lang.String pType,
  java.lang.String pDesc,
  java.lang.String pCreatedBy,
  java.lang.String pCreatedDate,
  java.lang.String pLastAccess)
- ***getDocument****(java.lang.String pName)* returns an RLVDocument object which includes document name, type, description, owner, created date, and date of the last access.
- ***deleteDocument****(java.lang.String pName,*
  java.lang.String ptype,
  java.lang.String pDesc) returns a Boolean which indicates if the transaction is successful.

3. Data Attribute:

com.nasa.rlv
Class RLVData

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVData
```

RLVData is the interface for data attribute where it can tell the relationships or references of the object to other objects in the database schema. *RLVData* interface has setter/getter methods to access to its attributes, which are concept name, vehicle name, mission name, trajectory name, discipline name, analysis name, and any runs that are available for the object (e.g., getAnalysis(), getMission(), getConcept(), getVehicle(), etc).
To access to an object *data attribute*, one can use:

- ***setDataValue****(java.lang.String[] dataValues)* where dataValues is an array that includes other references to the object (e.g. concept, vehicle, mission, trajectory, discipline, analysis, or run)
- ***getDataList()*** returns vector of RLVData
- ***getData****(java.lang.String pRName)* returns an RLVData object.
- ***deleteData****(java.util.Vector data)* returns Boolean which indicates if the transaction is success.

4. Geometry Attribute:

com.nasa.rlv
Class RLVGeometryFile

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVGeometryFile
```

RLVGeometryFile is the interface for geometry attribute where name serves as a key. *RLVGeometryFile* interface has setter/getter methods to access to its attributes, which are geometry file name, geometry type (e.g. iges, xyz, etc.), description, created date, directory path, and machine/host name. The geometry files associated to an object are stored in a file system, and an object can point to its geometry via metadata that stored in the database.

To access to an object geometry attribute, one can use:

- ***setGeometryFile****(java.lang.String pName,*
  java.lang.String pType,
  java.lang.String pDesc,
  java.lang.String pCreatedDate,
  java.lang.String pPath,
  java.lang.String pLocation)
- ***deleteGeometryFile****(java.lang.String pName,*
  java.lang.String ptype,
  java.lang.String pDate) returns Boolean which indicates if the transaction is success.

- ***getGeometryFile****(java.lang.String pName)* returns a *RLVGeometryFile* object
- ***getGeometryFiles()*** returns a vector of all *RLVGeometryFile* objects.

5. Notes Attribute:

com.nasa.rlv
Class RLVNotes

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributePrimitive
              |
              +--com.nasa.attribute.AttributeString
                    |
                    +--com.nasa.rlv.RLVNotes
```

RLVNotes is the interface for notes attribute, which acts as a white board for the object. To access to an object notes attribute, one can use:

- ***getNotes()*** returns RLVNotes object
- ***setNotes****(RLVNotes pNotes)* where pNotes is string.

4.5.1.3 User Account:

1. RLVAccount

com.nasa.rlv

Class RLVAccount

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVAccount
```

Direct Known Subclasses:

User

RLVAccount interface includes the following attributes: account, email, name, organization, and phone. Setter/Getter methods are used to access to RLVAccount attributes.

2. RLVAuthorization

com.nasa.rlv

Class RLVAuthorization

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeUser
                    |
                    +--com.nasa.rlv.RLVAuthorization
```

RLVAuthorization interface keeps track of user's permission, group, password, and login name.

3. ***User***

com.nasa.rlv

Class User

```
java.lang.Object
  |
  +--com.nasa.attribute.Attribute
        |
        +--com.nasa.attribute.AttributeComplex
              |
              +--com.nasa.attribute.AttributeID
                    |
                    +--com.nasa.rlv.RLVAccount
                          |
                          +--com.nasa.rlv.User
```

User interface is a subclass of the *RLVAccount* interface.

4.5.1.4 Conclusion:

The above list is the main checked points of the current ObjectStore database. ODBMS has some advantages. It allows a more natural modeling of complex data and couples the data and application logic. It leverages object oriented development model in term of inheritance, polymorphism, and encapsulation for the code and data. However, as the data grow more complex, a better algorithm to build the data root is required to reduce the data retrieving time.

4.5.2 Other possibilities RDBMS

Relational Database is a set of tables, which represent data in a two dimensional form. Data has no inheritance, polymorphism, or encapsulation. It is easy to understand the relationship between data and metadata. In relational model, logic is decoupled from data. Unlike object database, which presents data in a multiple dimensional form, relational database schema will be clearer and straightforward. This will make the development of the business logic application easier and less messy.

The transition between ODB and RDB is to map object to relational. The mapping will add a translation layer between object state and relational database, which is tables. Therefore, sophistication level of mapping can be directly limiting to the object model.

Relational Database has been dominant in the market. In addition, there are many vendors to choose from, and it makes sharing data easier.

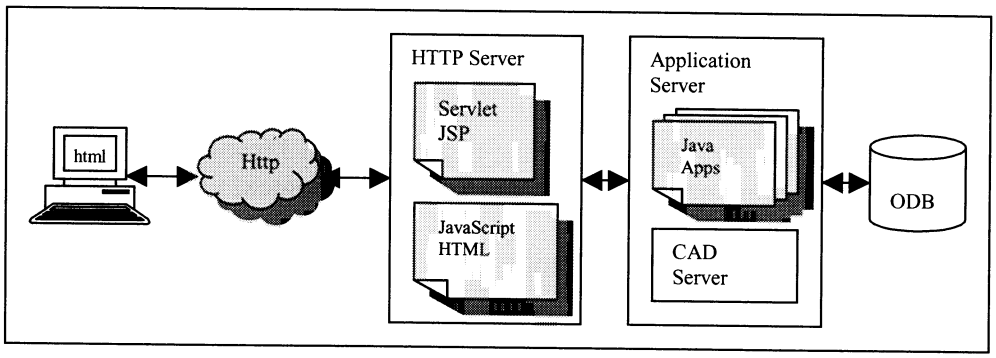
5.0 System Architecture

5.1 adttWeb Overview

The *adttWeb* is a multi-tier Client/Server architecture where the user can log in from a browser via the URL:

http://www.nas.nasa.gov:8080/adttWeb

When logging in, user needs to provide a login name and a password. The system will authenticate the user and give the user the connection to the system application and database.



The *adttWeb* project has adopted the new technologies for the e-Business model. Written in pure Java from front to back end, it guarantees the platform independent, server independent, and the system scalability and mobility. Servlet and JSP pages are used to dynamically generate HTML documents instead of CGI because servlets/JSP pages don't start new process for each request, they can run in the same server process as HTTP server, and they are multi-threaded. Furthermore, since CGI is written in C/C++, CGI web server is a platform dependent application.

Back end application server provides the connectivity to the database and application logic. Application logic includes java applications, CAD server, and Data repository Server. Middleware includes RMI and HTTP protocols.

Notes:

Microsoft Internet Explorer 5.5 or later is recommended to capture the geometry and plot view since third party vendor are using ActiveX and Visual Basis for displaying the graphical contents.

5.2 adttWeb Integration

5.2.1 The adttWeb Web Server

- *Apache Linux* is an existing web server supported by NAS Web Group.
- *Tomcat Server* is a servlet and JSP engine has been installed into Apache web server per adttWeb project request.

- Programs that drive the adttWeb include Java applications, Servlet, JSP pages, HTML documents, JavaScript, and Visual Basic Script.
- Java RMI *(Remote Method Invocation)* is used as middleware that connects the web server and application server.

Currently, the Tomcat Server resides on *taco.nas.nasa.gov* machine and the *hostname* is *www.nas.nasa.gov:8080*. Since the Tomcat Server or Tomcat Servlet engine is pure java, it becomes OS and platform independent. The software is an open source program and free for downloading hosted by *apache* (http://www.apache.org). Therefore, the *adttWeb* website embedded in Tomcat environment can be moved to any machine as long as the web server accessible through firewall. The *hostname* might be changed accordingly to the web server host machine name.

According to Tomcat Servlet engine, all of the html, JSP, and JavaScript files must be placed under Tomcat document base structure. To set up a context path, a new Context object must be added in:

/TOMCAT_HOME/conf/server.xml

The file structure for the Tomcat Servlet engine is set as follow:

- /TOMCAT_HOME/webapps/adttWeb/: contains all of the html, JSP, and JavaScript files
  - /TOMCAT_HOME/webapps/adttWeb/AddDelete
  - /TOMCAT_HOME/webapps/adttWeb/Analyze
  - /TOMCAT_HOME/webapps/adttWeb/TabData
  - /TOMCAT_HOME/webapps/adttWeb/TabDocuments
  - /TOMCAT_HOME/webapps/adttWeb/TabHistory
  - /TOMCAT_HOME/webapps/adttWeb/TabNotes
  - /TOMCAT_HOME/webapps/adttWeb/TabParameters
  - /TOMCAT_HOME/webapps/adttWeb/TabStudies
  - /TOMCAT_HOME/webapps/adttWeb/TabViews

- /TOMCAT_HOME/webapps/adttWeb/images/: contains all of the images necessary for the website.
- /TOMCAT_HOME/webapps/adttWeb/script/: contains all of the java scripts that support the website.
  - *loadtree_analysis.js:* contains the database contents of Project/Discipline objects for the navigation tree. This is a runtime process if there is any update action to the database.
  - *loadtree_concept.js:* contains the database contents of Project/Concept objects for the navigation tree. This is a runtime process if there is any update action to the database.
  - *loadtree_mission.js:* java script contains the database contents of Project/Mission objects for the navigation tree. This is a runtime process if there is any update action to the database.

- *scriptToolBox.js:* contains functions for different styles of popup windows.
- *trajectory.js:* supports the trajectory viewer including animation, mouse driven events, and loading bitmap images.
- *treescript.js:* contains functions that support different objects for the navigation tree.

§ /TOMCAT_HOME/webapps/adttWeb/WEB-INF/classes/: contains all of the servlet and java classes.

- *AccountAdministrator.java*: is a part of RMI client stuff that authenticates user, creates new account, queries an existing account, and updates an existing account.
- *DataAccessClient.java*: is a part of RMI client stuff that provides full access to the database (e.g. queries object attributes, updates object attributes, deletes object attributes)
- *HavocDataFile.java:* extracts data from output file of Havoc run (e.g. WINGBODY.plt)
- *NewUserFormBean.java*: is a java bean that validates new user input form and also checks if new user input login name already existed in the database (if the login name already existed, user must choose a new user login name).
- *Plot2dInfo.java*: is a temporary java bean that keeps information (e.g. legend names, name of Havoc files to be plotted) for 2D plotting process in the memory during the user's session
- *UploadFile.java/UploadFile2.java:* is a servlet that transforms uploaded document into stream of bytes, which will be saved in the database as blobs.
- *UserLoginInfo.java*: is a temporary java bean that keeps a user's information (e.g. login name, password, session id, user name, the authorized data source) for the entire user's session.
- *UserToolBox.java*: is a helper java class of reusable components (e.g. parser functions, mapping object mechanism)

5.2.1.1 Create New Accounts

File/Directory structure:
/TOMCAT_HOME/webapps/adttWeb/admin/

Creating a new account for a new user is an administrator's task and can be done via either command line java application or *adttWeb* website. To access to *Administrator pages,* one can log in to:
http://www.nas.nasa.gov:8080/adttWeb
Click on: *Login* hyperlink
Click on: 'New User? Please Click Here' hyperlink
The *Administrator pages* has the following tasks:

- CREATE ACCOUNT
  - User Account: fully implemented
  - Admin Account: fully implemented
- QUERY
  - User Account: not implemented since it's not the task priority
  - Admin Account: not implemented since it's not the task priority
- UPDATE
  - User Account: not implemented since it's not the task priority
  - Admin Account: not implemented since it's not the task priority
- DELETE
  - User Account: not implemented since it's not the task priority
  - Admin Account: not implemented since it's not the task priority
- LOGOUT: fully implemented.

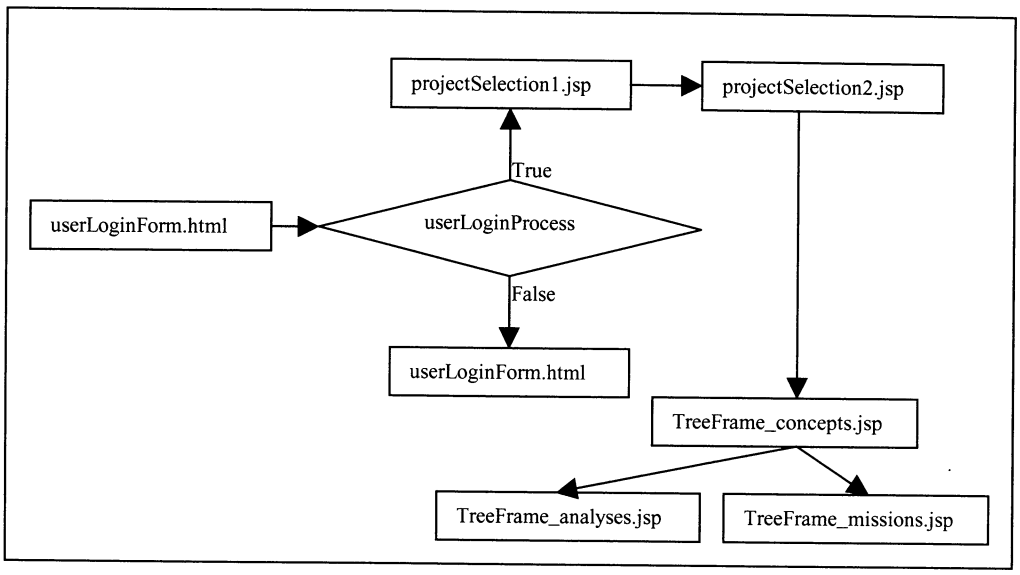
Currently, new account request is done by phone or email. The *adttWeb* administrator will create a new account with the required information from the user:

- Name: user's full name
- Email: user's email address
- Phone: user's telephone number
- Organization: user's organization (e.g. branch, company, division, etc)
- Login Name: user's last name (in lower case by default)
- Password: is user's login name (user's is advised to change the password at the first time logging in and not to choose sensitive password since the authenticate system is not fully encrypted)
- Group: group number or group name. (This field is used to set user's permission to log in into certain project. Currently not implemented)

Once the new account created, the administrator will email to the new user the login name and password. Since this is not the priority of the project, administrator has to reply new account via manually emailing system. This process can be automated using JavaMail for the system enhancement.

5.2.1.2 User Log-on

User log-on process requires the user to enter user's login name and user's password. The flow chart of the user log in process can be described as the following:



File/Directory structure:

/TOMCAT_HOME/webapps/adttWeb/userLogin/

The *userLoginProcess.jsp* collects the user's login name and password and sends the login information to application server where the system authenticates the user against the user account information stored in the system database. If the user is not authorized, the system will throw the user back to the *userLoginForm.html,* else the system will grant the user a session id and forward the *projectSelection1.jsp* page where user can choose a project. The *projectSeclection2.jsp* will forward the user to the *TreeFrame_concepts.jsp* with the appropriate navigation tree according to the chosen project.

Currently, the system does not use cookies to keep track the user's activity. Instead, it uses the user's granted session id. The session id acts as a pass ticket as the user navigates through the system. The session id time out (i.e. expires if there is no activity in a period of time) can be set by the system developers. The session id will be invalidated when the user logs out of the system.

5.2.1.3 User Log-out

If user currently owns a session, logging-out will invalidate the user's session. User needs to log in again in order to navigate the site.

5.2.1.4 Web Applications

The default navigation tree is the Project/Concept objects tree. The tree appears on the left frame of the page. There are also hyperlinks (concepts, Missions, and Analyses) that allow user to click to get the new navigation tree or refresh the existing navigation tree.

The navigation tree provides a full view of the *Project/Concept*, *Project/Analysis*, or *Project/Mission* objects in the database. The tree presentation is a JavaScript implementation, which includes folder nodes and link nodes. A link node presents an object (e.g. *Concept, Vehicle, Subassembly, Part, Discipline, Analysis, Run, Mission, Trajectory, TrajectoryPoint)* in the database. Clicking on the link node will give the user information about that object (e.g. Data, History, Notes, Parameters, Views, Documents, Studies, Analyze, and Add/Delete).

- **Data**
  - URL: http://hostname/adttWeb/TabData/objectData.jsp
  - Click on *Data* tab. Image source: /adttWeb/images/tab/data.gif
  - Displays a table of inter-relationship of the object with other object in the database: CONCEPT, VEHICLE, MISSION, TRAJECTORY, DISCIPLINE, ANALYSIS, RUNS.
  - Functions: *selects* and *views* 2-D plot of the object's analyses runs.
    - */adttWeb/ViewPlot/choosePlot.jsp*: allows user to select one mission phase per run to plot
    - */adttWeb/ViewPlot/viewPlot.jsp*: has top and bottom frames
      - */adttWeb/ViewPlot/viewplotTop_displayplot.jsp*: allows user to customize the look-and-feel of the plot at run time.
      - */adttWeb/ViewPlot/viewplotBottom_displayplot.jsp*: displays the customized plot.
    - *legendinput.jsp* and *legendinput_process.jsp*: helper pages to customize the legend labels.

- **History**
  - URL: http://hostname/adttWeb/TabHistory/objectHistory.jsp
  - Click on *History* tab. Image source: /adttWeb/images/tab/history.gif
  - Displays the hierarchical relationship of the object in the database.

- **Notes**
  - URL: http://hostname/adttWeb/TabNotes/objectNotes.jsp
  - Click on *Notes* tab. Image source: /adttWeb/images/tab/notes.gif
  - Displays the white-board notes of the object.
  - Functions: User can *edit* the notes and save it into the database *(/adttWeb/TabNotes/updateNotes.jsp)*

- **Parameters**
  - URL: http://hostname/adttWeb/TabParameters/objectParameters.jsp
  - Click on *Parameters* tab. Image source: /adttWeb/images/tab/para.gif
  - Displays all of the parameters of the object. Parameter has six fields: name, value, units, description, type, code variable, and changeable flag. If changeable is TRUE, then the parameter attributes are displayed in text fields, and user can modify the field value. If changeable is FALSE, then the parameter will be displayed as hypertext, and user cannot modify the field value.
  - Functions:
    - *modifyParameters.jsp*: User can *modify* the parameter attribute field and save the new modification as under the current object, a new vehicle where new vehicle name must be provided, or a new version of the current vehicle.
    - *addParameter.jsp*: User can *add* a new parameter to the parameter list of the object.
    - *DeleteParameter.jsp*: User can delete an existing parameter for the parameter list of the object.

- **Views**
  - URL: http://hostname/adttWeb/TabView
  - Click on *Views* tab. Image source: /adttWeb/images/tab/views.gif
  - Displays panel that allow user to choose to view the geometry via Vuent/iEngineering collaboration tools or view the object trajectory via WildTangent tools. The view user interface panel provides different choices depending on the level of the object. At Vehicle level or below, the view displays the active object. At the concept level or above, the view displays the comparison views between vehicles.
  - Functions:
    - */adttWeb/ViewTrajectory/trajectory2.htm:* a JavaScript that displays the 3D automation of a vehicle trajectory (*jeffx33.wt*)
    - */adttWeb/TabView/viewGeometry.jsp:* User can choose to view comparison view between vehicles or to view a particular vehicle or object (depend on the active object level.)
    - */ViewGeometry/viewGeometryViaVuent.jsp:* Displays the object geometry via Vuent collaboration tools. The program activates the vehicle geometry am3 file that links to Vuent geometries resided on the CAD server (*Gomez). Vuent API* VisualBasic script allows the program to display the whole vehicle or a particular part geometry.
    - */adttWeb/TabView/viewGeometryOptions.jsp:* Displays two dropdown list of vehicles where user can choose a vehicle from each list.
    - */ViewGeometry/comparingGeometriesViaVuent.jsp:* Displays the two chosen vehicles for comparison side by side.

- **Documents**
  - URL: http://hostname/adttWeb/TabDocuments/viewDocuments.jsp
  - Click on *Documents* tab. Image source: /adttWeb/images/tab/docs.gif
  - Displays the object related documents. Document list is the technical documentations associated with the object that user uploads to the database. Document attributes have six fields: name (once the document uploaded it will be assigned a unique name which is objectName_randomIndex.fileExtension), type (file extension e.g. doc, pdf, gif, jpg, txt, etc.), owner (user login name of whom uploads the document), created date (the date where the document is uploaded), and last accessed date (the date where document has been viewed).
  - Functions: User can *View* all of the documentation available of the object. The document name is a hyperlink to actual document file in the web server; therefore, user can download the shared document using right mouse button.
    - */adttWeb/servlet/UploadFile2.java:* This servlet linked to the 'Upload' button that allows user to upload a file from the user's local drive to the object's document list in the database.
    - */adttWeb/TabDocuments/deleteDocuments.jsp*: This JSP page linked to the 'Delete' button that allows only user who owns the document to delete the document from the object's document list in the database.
- **Studies**
  - URL: http://hostname/adttWeb/TabStudies/studies.jsp
  - Click on *Studies* tab. Image source: /adttWeb/images/tab/studies.gif
  - This feature has not yet implemented since there is no exact requirements and guidelines provided.
- **Analyze**
  - URL: http://hostname/adttWeb/TabAnalyze/analyze.jsp
  - Click on *Analyze* tab. Image source: /adttWeb/images/tab/anal.gif
  - Displays Analysis and Mission dropdown list where User can choose certain disciplines to run an analysis. The contents of the analysis and mission list are extracted from the database under Project/Discipline/Analysis and Project/Mission/Trajectory objects.
  - Functions:
    - */adttWeb/TabAnalyze/analyzeProcess.jsp*: This JSP page linked to the 'Analyze Vehicle' button where use can submit an analysis run job under the vehicle object. Currently available is HAVOC analysis.
- **Add/Delete**
  - URL: http://hostname/adttWeb/TabAddDelete/addDeleteElement.jsp
  - Click on *Add/Delete* tab. Image source: /adttWeb/images/tab/addel.gif
  - Allows user to *add* a children object, *delete* a current object, or *rename* a current object.

- Functions:
  - *addElement.jsp*: This JSP page displays a form where user needs to enter a name for the children object to be added
  - *addElementProcess.jsp*: This JSP page linked to the 'Add' button, which submits a request to application server to add a new object to the database.
  - *deleteElement.jsp:* This JSP page display a form to get the confirmation from the user to delete the current object and all of its children.
  - *deleteElementProcess.jsp*: This JSP page linked to the 'Delete' button, which submits a request to the application server to delete the object recursively from the database.
  - *renameElement.jsp:* This JSP page displays a form where user needs to enter a new name for the object rename.
  - *renameElementProcess*: This JSP page linked to the 'Rename' button, which submits a request to the application server to rename the current object name in the database.

5.2.1.5 Web Data Repository:

File/Directory structure:

/TOMCAT_HOME/webapps/adttWeb/webData/

This is a temporary container that supports the website data at runtime for viewing HAVOC output results, plotting 2D graph, viewing object geometry via Vuent, and downloading object documents. It contains the following directories:

- /HAVOC: is where the output and input HAVOC run analysis must be forced to the web server outside of the NAS firewall so that user can view the input and output results via the *adttWeb* website.
- /UPLOADS: is a temporary disk space at runtime used in uploading documents of the object to the database. This directory is subject to clean at any time.
- /am3: is where the Vuent descriptor files being forced to the web server outside of the NAS firewall so that user can view the object's geometry.
- /downloadFiles: is a temporary disk space at runtime used in downloading documents of the objects from the database. This directory is subject to clean at any time.

Web Document:

File/Directory structure:

/TOMCAT_HOME/webapps/adttWeb/doc/

This directory contains static HTML pages associated to the *adttWeb* website.

Web Error Document:

File/Directory structure:

/TOMCAT_HOME/webapps/adttWeb/error/

This directory contains static HTML pages posting error message for the *adttWeb* website.

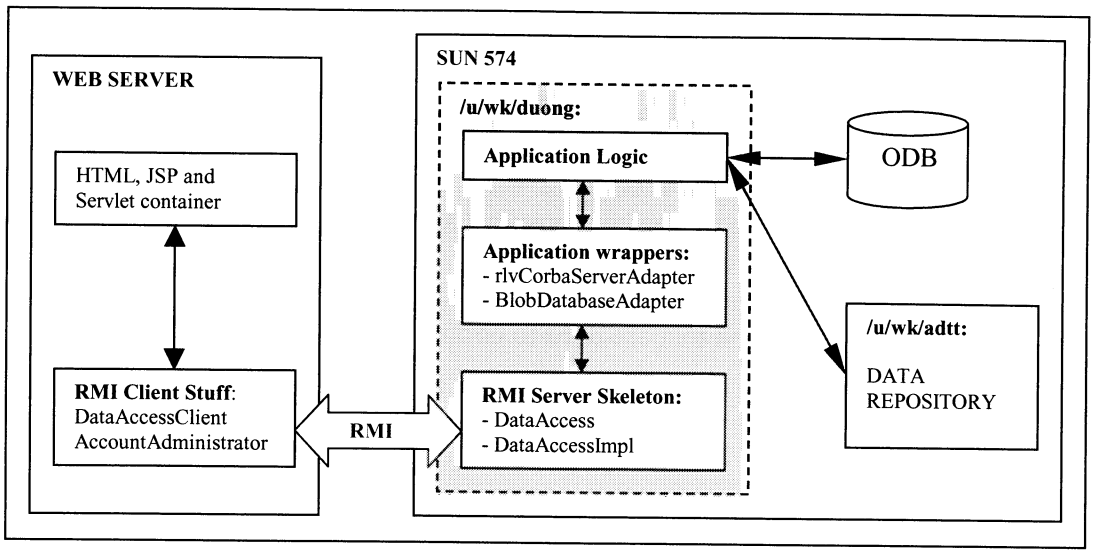
Web Help Document:
File/Directory structure:
/TOMCAT_HOME/webapps/adttWeb/helpDoc/
This directory contains static HTML pages posting help message for the *adttWeb* website.

5.2.2 Application Server

The application server resides on *Sun574.nas.nasa.gov* machine. It includes java source code, java classes, other data repository, and temporary working disk space to support running analysis job.

The system spans two user directories on Sun574: */u/wk/adtt* and */u/wk/duong*.

- *System Data Repository* occupies heavily on */u/wk/adtt/* directory where the permission is set to user/group read and write able. It contains the data to support the application logic.
  - */BOM*: stores the ProE objects in text file for uploading the objects to the database.
  - */attributes*: stores text files of object parameter list for uploading/downloading to the database via java applications.
  - */fortran:* executes sniffer programs to update the web contents at runtime
  - */geometry*: store ProE and other CAD software geometry files (e.g. igs, xyz)
  - */havoc/havocscript:* contains runHavoc.sh that created/updated at runtime for HAVOC analysis job.
  - */havoc/havocinput*: temporary container of input data for HAVOC analysis job.
  - */havoc/havocoutput*: temporary container of in/out data for HAVOC analysis job.
  - */webapps/webData*: temporary container used to force the webData content to the web server.
- *Java RMI* is the connection between the web server and sun574 through the NAS firewall.



- The *Application Logic* contains java classes for the business logic and the database connectivity. It also consists of a number of C-shell scripts to support analysis job launching, updating database contents to the JavaScript programs used in displaying the navigation trees for the website.
    - Home Directory structure for the database connectivity:
      /u/wk/duong/RLVjava2/RLVCorbaServer/com/nasa
      It contains java packages: *attribute, database,* and *rlv.*
    - Home Directory structure for the application logic:
      /u/wk/duong/RLVjava2/RLVCorbaServer
      It contains several java packages: *analysisCodes, communication, rmiDataAccess, CorbaServer, CorbaServer.accounts, CorbaServer.loaders,* and *CorbaServer.uploadDatabase.*
    - Java package definitions:
        - *analysisCodes*: package contains java classes supporting the analysis code runs (e.g. *HavocHelper)*
        - *communication*: package contains java classes to make connection to other servers (e.g. connection to CAD Server via HTTP protocol)
        - *CorbaServer*: package contains java classes for the business logic such as data access and data manipulation (e.g. *QueryRLVDataList, QueryRLVParameters, CreateNewVehicle, etc)*
        - *CorbaServer.accounts*: package contains java classes that support user's accounts including authentication, update, or create new account.
        - *CorbaServer.loaders*: package contains java classes and java applications that upload data to the database or download data

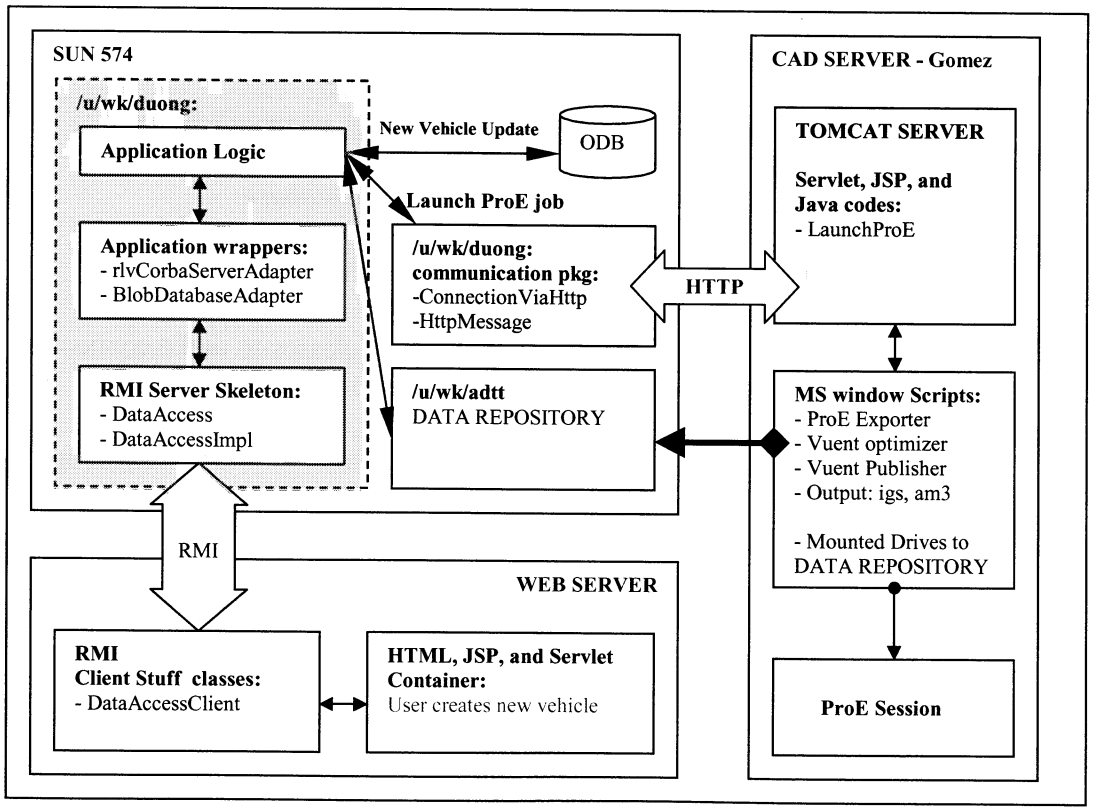
from the database to build JavaScript navigation trees or to a text file.

- *CorbaServer.uploadDatabase*: package contains java applications that initialize the database or upload objects to the database from ProE BOM text file.
- *test*: package contains java applications that are used for testing java modules.
- *rmiDataAccess*: package contains java classes of the RMI server skeleton that are used to communicate with RMI clients.

5.2.3 CAD Server

The CAD server resides on *Gomez.nas.nasa.gov* window NT 4.0 machine. It includes a Tomcat HTTP server with JSP, servlet, and java codes, MS window scripts (i.e. *.bat), and temporary working disk space to support creating new geometry job.

The scenario is when user clicks on the button to create a new vehicle or a new version of the current vehicle, the request will be sent and executed on the application logic. The job is to create a new vehicle object in the database and launch the ProE job to generate a new vehicle per requested information.



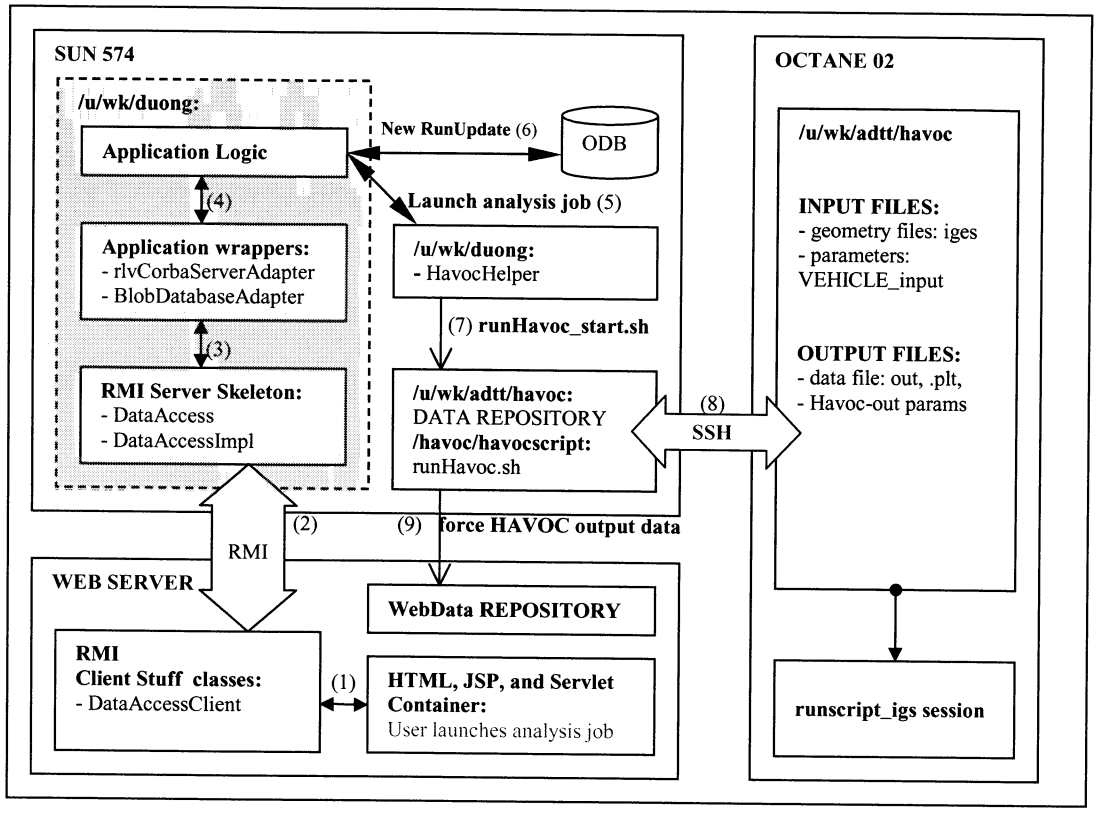
- HTTP Server – tomcat is installed to Gomez to launch the command to start the ProE job. The reason to use HTTP server as the middle ware and window NT as the CAD server is that ProE has floating licenses that can be run on any machines (e.g. SunOS, SGI, NT), but Vuent software only can be on window NT. Other alternative middle wares can be used such as CORBA and RMI, but those require more programming overhead. Therefore, we have to choose window NT as the CAD Server.
- MS-dos scripts, Java applications, and Servlet pages.
    - URL: http://gomez.nas.nasa.gov:8080/adttWeb-support/servlet/LaunchProe
    - File/Directory structure on the CAD server:
      \TOMCAT_HOME\webapps\adttWeb-support\WEB-INF\classes
    - Functions:
        - § Parses the input stream of parameter list and new vehicle information into input data files necessary for ProE run.
        - § Prepares veb file and creates directories necessary for Vuent Optimizer.
        - § Launches the system MS window script to run ProE
          E:\bin\bat_killer1.bat
        - § Launches the system MS window script to move ProE output iges files to the DATA REPOSITORY and to run ProE Exporter to get the 3dx output files.
          E:\bin\bat_killer2.bat
        - § Launches the system MS window script to run Vuent Optimizer to create am3 files and Vuent Geometries necessary for Vuent Viewer.
          E:\bin\bat_killer3.bat

5.3 Run Analysis Code

Currently, there is only HAVOC analysis code that has been fully implemented and passed the flow tests. The HAVOC analysis program is written in Fortran resides on *Octane02.nas.nasa.gov* SGI machine.

File/Directory structure:
/u/wk/adtt/havoc
This directory is a temporary disk space container used to run HAVOC analysis code. The command starts with:
/u/wk/adtt/havoc/runscript_igs VEHICLE_input

The scenario is when user clicks on the button to run an HAVOC analysis job of the current vehicle, the request will be sent and executed on the application logic. The job is to retrieve information necessary of the vehicle, create the environment for the analysis run, and launch the analysis run to *Octane02*.



There are sequences of processes happening when an analysis job is submitted.

- (1) User launches an analysis job from URL:
  http://host/adttWeb/TabAnalyze/analyze.jsp
- (2) The analysis job will be sent to the application server (sun574) using RMI client stuff classes: *DataAccessClient*, which will marshal the message and send it to RMI server.
- (3) The RMI server skeleton will un-marshal the message and call the application wrapper, *rlvCorbaServerAdapter*
- (4) *rlvCorbaServerAdapter,* which invokes the *CorbaServer.RunAnalysisProcess* class. This class is responsible for the steps: (5) and (6)
- (5) Using *analysisCodes.HavocHelper* to prepare the environment for the HAVOC run and launch the analysis job:
  - Prepares the vehicle geometries (iges) to the havoc temporary input directory: */u/wk/adtt/havoc/havocinput.*
  - Extracts vehicle parameter list from the database and writes it to the text file in the */u/wk/adtt/havoc/havocinput* directory (e.g. WINGBODY_params.dat).
  - Generates the */u/wk/adtt/havoc/havocscript/runHavoc.sh* script with current run information.
  - Creates all the directories necessary for storing output data file after the run finishes.

- If all the above tasks in step (5) success, then this will invoke a static system shell script to start the */u/wk/duong/RLVjava2/RLVCorbaServer/script/runHavoc_start.sh*

- (6) Adds the new Analysis and Trajectory information of the run to the vehicle and add new run object to the database
- (7) When the */u/wk/duong/RLVjava2/RLVCorbaServer/script/runHavoc_start.sh invoked at the end of step (5).* This script changes the mode permission in the running HAVOC directory and invokes the */u/wk/adtt/havoc/havocscript/runHavoc.sh* script.
- (8) *runHavoc.sh* calls Perl script parser to parse the data input file, does the 'scp' the input files into generic files on Octane02, and 'ssh' to Octane02 temporary Havoc run directory: */u/wk/adtt/havoc*. It will launch the script, *runscript_iges.sh*. After the run finishes, *runHavoc.sh* will 'scp' back the output result files to the DATA REPOSITORY on Sun574, parse the output HAVOC-out parameters, and upload the HAVOC-out parameters to the database.
- (9) A FORTRAN program, *sniffer*, will force the output data to the web server so that user can view or download the run output results.

5.4 Conclusion and Recommendations

Generally in house java implementation of an application servers lack transaction management, security, concurrency, load balancing, and connection pooling. The current system only allows a small number of users. It depends on the built-in lock mechanism of the database vendor to manage the transaction.

For a large number of users and system sufficiency, we should consider purchasing an application server from the COTS tools so that it can simplify application development. Developers should only concentrate on the application logic, and let the COTS tools to handle the complexity of lower level services. In addition, Sun provides a free download of an application server, J2EE, and we should consider trying it on top of the back end database and application logic.

Currently, the database and the application logic are in /u/wk/duong directory due to this still being in the development stage. This could be moved to /u/wk/adtt or to any other machine.

6.0 System Networking and Security

The *adttWeb* system is a multi tier Client/Server system. In order for the Internet user who is outside of the NAS firewall or NAS intranet logs into the system, the system web application must be on the web server that is outside of the NAS firewall. This raises many issues such as how to protect the *ADTT* data without violating NAS security policies.

There are four plans proposed for the *adttWeb* system networking and security. In all of the four plans for the proposal, the Internet user must log into Taco (a NAS web server outside the NAS firewall), and the *index.html* of the *adttWeb* system must reside on Taco.

Since the *adttWeb* website implement the *apache-Tomcat servlet engine*, all of the *adttWeb*'s HTML, JSP, and servlet pages will be in the Tomcat docBase context. Currently, it is on Taco.

The Internet user who would like to log into the *adttWeb* logs into the URL:
http://www.nas.nasa.gov:8080/adttWeb

6.1 Plan 1

This plan proposes to use two web servers: one is outside of the firewall and one is inside of the firewall.

- *(Please see figure 6.6-1)*
- Taco, the external web server, which is used for querying data from the database. Connection service is **RMI** between Taco and application database server.
  http://www.nas.nasa.gov:8080/adttWeb

    - Sun574, the internal web server, which is used for viewing geometries and plots. The viewing request from Taco will be routed to Sun574 web server via **HTTP node-locked** which only allows access from Taco to designated internal server

  http://sun574.nas.nasa.gov:8080/adttWeb-support

a. Implementation

- Internet user logs on to Taco from the URL:
  http://www.nas.nasa.gov:8080/adttWeb
  Internet user logged in from Taco use RMI connection to retrieve and update the data from the database. All of the data include database, geometry files, and data analysis input and output files are resided on internal machine (e.g., sun574).

- The JavaScript files containing database objects that used to display the navigation trees will be updated per request and will be 'force' to Taco, the external web server. The data is being 'force' from sun574 machine:
  /u/wk/adtt/jakarta-tomcat/webapps/adttWeb-support/script
  To build-dev machine:
  /www/science/htdocs/codeA/adtt/webapps/adttWeb/script

- When user requests to view geometries, analysis data, and related documents from the internal file system, Taco will route the request to a designated internal web server. In this case, the internal web server is sun574 at the following URL:
  http://sun574.nas.nasa.gov:8080/adttWeb-support/ViewPlot
  http://sun574.nas.nasa.gov:8080/adttWeb-support/ViewGeometry
  http://sun574.nas.nasa.gov:8080/adttWeb-support/viewDocument.jsp

b. Notes

This is a **wish-plan** since all the data are resided on internal machines. When users request to access to files or data from database, user will be routed to the right web server transparently, and the present data is in real time.

However, there are several concerns:

- RMI is a key player or, in other words, is an oxygen-life-support mechanism for the whole system. NAS security policy must allow a certain port open for this connection
- NAS security policy must allow implementing the HTTP node-locked mechanism to a designated machine.
- Still have some delay to refresh data from database, especially, if there is new object such as new vehicle, new version, new part, etc has been added to the database. This might cause uncomfortable to the user.

6.2 Plan 2

This plan proposes to use only one web server, which is outside of the firewall. This web server also acts as a web data repository.

- *(Please see figure 6.6-2)*
- Taco, the external web server, which is used for querying data from the database. Connection service is **RMI** between Taco and application database server.
  http://www.nas.nasa.gov:8080/adttWeb

c. Implementation

- Internet user logs on to Taco from the URL:
  http://www.nas.nasa.gov:8080/adttWeb

Internet user logged in from Taco use RMI connection to retrieve and update the data from the database. All of the data include database, geometry files, and data analysis input and output files are resided on internal machine (e.g., sun574). However, the geometry files, data analysis input and output files, and downloadable documents from the database needed to present on a web server for viewing and plotting. These set of data needed to be 'force' to the external web server every time the data updated.

- The JavaScript files containing database objects that used to display the navigation trees will be updated per request and will be 'force' to Taco, the external web server. The data is being 'force' from sun574 machine:
  /u/wk/adtt/jakarta-tomcat/webapps/adttWeb-support/script
  To build-dev machine:
  /www/science/htdocs/codeA/adtt/webapps/adttWeb/script

- Viewing plot, view geometry, and view documents features are in the external web server.
  http://www.nas.nasa.gov:8080/adttWeb/ViewPlot
  http://www.nas.nasa.gov:8080/adttWeb/ViewGeometry
  http://www.nas.nasa.gov:8080/adttWeb/viewDocument.jsp
- When user requests to view geometries, analysis data, and related documents, the resources are available on the external web server. The data are at the web data repository, which is a mirrored copy of the internal file system. The data is being 'force' from sun574 machine:
  /u/wk/adtt/webapps/webData and
  /u/wk/adtt/jakarta-tomcat/webapps/adttWeb-support/downloadFiles
  To build-dev machine:
  /www/science/htdocs/codeA/adtt/webapps/adttWeb/webData

d. Notes

This is the **current-status** plan. Under this plan, RMI carries request from Taco to internal server. Internal server will process the request. When done, internal server will use **ssh** and **scp** to transfer files to build-dev and Taco, then the data will be 'force' to make it available to user. RMI is a key player or, in other words, is an oxygen-life-support mechanism for the whole system. This plan will raise some difficulties:

- The delta time from user's point-and-click to the data available on Taco point is at some small delay. User sometime needs to 'hit' the refresh button.
- Analysis and geometry data are exposed on the external machine.

6.3 Plan 3

This plan proposes to use only one web server, which is outside of the firewall. The database and data repository will be on a machine outside of the firewall. Also, the plan needs to have another data repository machine inside of the firewall.

- *(Please see figure 6.6-3)*
- Taco, the external web server, which is used for querying data from the database. Connection service is **RMI** between Taco and application database server.
  http://www.nas.nasa.gov:8080/adttWeb
- Cod, the external application server, data repository, and database.
- Sun574, the internal data repository server, is used to synchronize the data to the external data repository.

e. Implementation

- Internet user logs on to Taco from the URL:
  http://www.nas.nasa.gov:8080/adttWeb
  Internet user logged in from Taco use RMI connection to retrieve and update the data from the database. All of the data include database, geometry files, and data analysis input and output files are resided on the external machine (e.g., Cod). However, the geometry files, data analysis input and output files which are the output results of the geometry generation and analysis job executed on a set of internal machines. Those data need to be transferred to Cod using **ssh** and **scp**.
- The functions to view geometries, analysis data, and related documents will be moved to Taco along with all the analysis input and output data. Viewing data will be on Taco as the following URLs:
  http://www.nas.nasa.gov:8080/adttWeb/ViewPlot
  http://www.nas.nasa.gov:8080/adttWeb/ViewGeometry
  http://www.nas.nasa.gov:8080/viewDocument.jsp

- All of the database and java interface will be on Cod (a sun machine for web database – NAS). AdttWeb external application server (Cod) talks to its internal server only by ssh. Request commands (e.g. to run ProE, analysis codes, etc.) to internal server using ssh. Internal server then processes the requests and sends the requests to Octane02 or Gomez to execute the job. When done, the data sent back to internal server where it will be scp to the adttWeb external server (Cod)

f. *Notes*

This is a **never-wanted** plan because of several concerns:

- All data and database are resided on unprotected machine outside of the firewall.
- Cod must support individual user account in order to set **ssh** to send requested commands to internal server.

6.4 Plan 4

This plan is similar to plan 1. The proposal is to use two web servers: one is outside of the firewall and one is inside of the firewall.

- *(Please see figure 6.6-4)*
- Taco, the external web server, which is used for Internet user logging into the system

  http://www.nas.nasa.gov:8080/adttWeb

  - IN machine, the internal web server, which contains the whole *adttWeb* web application system. Proxy is used to route the logged in user to this machine. It totally transparent to the user.

g. *Implementation*

- Proxy between the external web server to the internal web server
- Tomcat Servlet engine with the *adttWeb* contents resides on IN server.
- Internet user does not see the IN Server; instead, user logs in into Taco via URL:

  http://www.nas.nasa.gov:8080/adttWeb
- The functions to view geometries, analysis data, and related documents will be on IN server along with all the analysis input and output data. Viewing data will be routed to IN server from Taco as the following URLs:

  http://www.nas.nasa.gov:8080/adttWeb/ViewPlot
  http://www.nas.nasa.gov:8080/adttWeb/ViewGeometry
  http://www.nas.nasa.gov:8080/viewDocument.jsp

h. *Notes*

This plan is the **wish-plan** because the proxy is transparent to the user. All of the project data are located on the machine that is inside of the firewall. This proposal seems to be the best solution since the data need to be present on the web server can be resided on the internal machine.

Sadly, this plan is rejected by the NAS Security Group, and the reason is opening a port for proxy through the firewall increases hacker risk.

6.5 Conclusion

The main purpose of the proposals is to make sure that the ADTT web project complies with the NAS security policy.

Since NAS does not implement the VPN (Virtual Private Network), all of the users whose networks do not belong to the NAS network cannot log into the *adttWeb* system. The only solution is to put the websites on the external web server. This raises some difficulties such as how to extract the data from the internal machine to the web server, how to launch a job from an external machine to the internal machines, how to share the data among the authorized Internet users and how protect the data integrity at the same time, amongst other things.

Tomcat Servlet Engine and Java web applications are totally new to the NAS web server group. Incorporating knowledge and experience is important. This will help to expedite the process of bringing the *adttWeb* website to the respective users.

6.6 Diagrams

Figure: 6.6-1

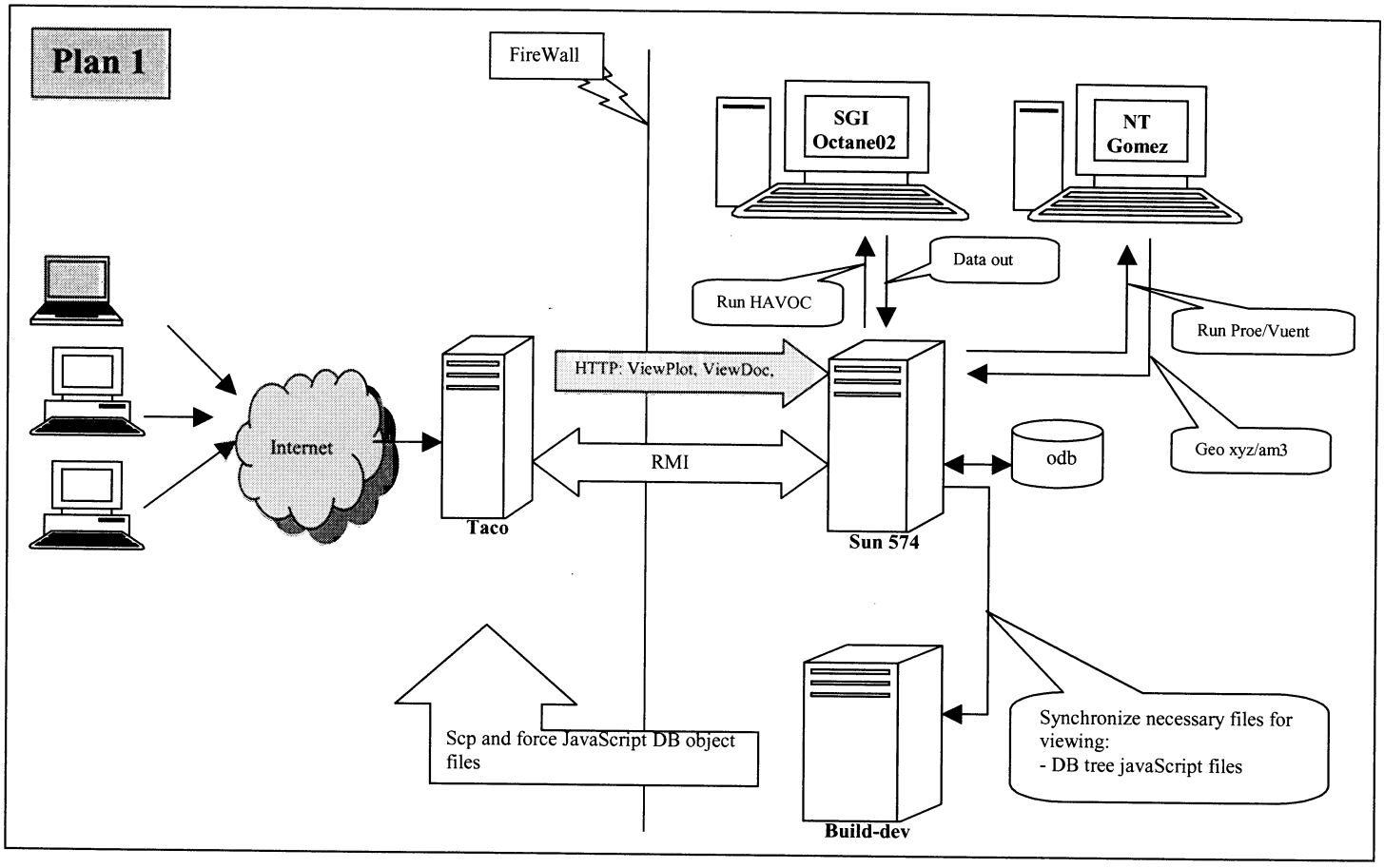


Figure: 6.6-2

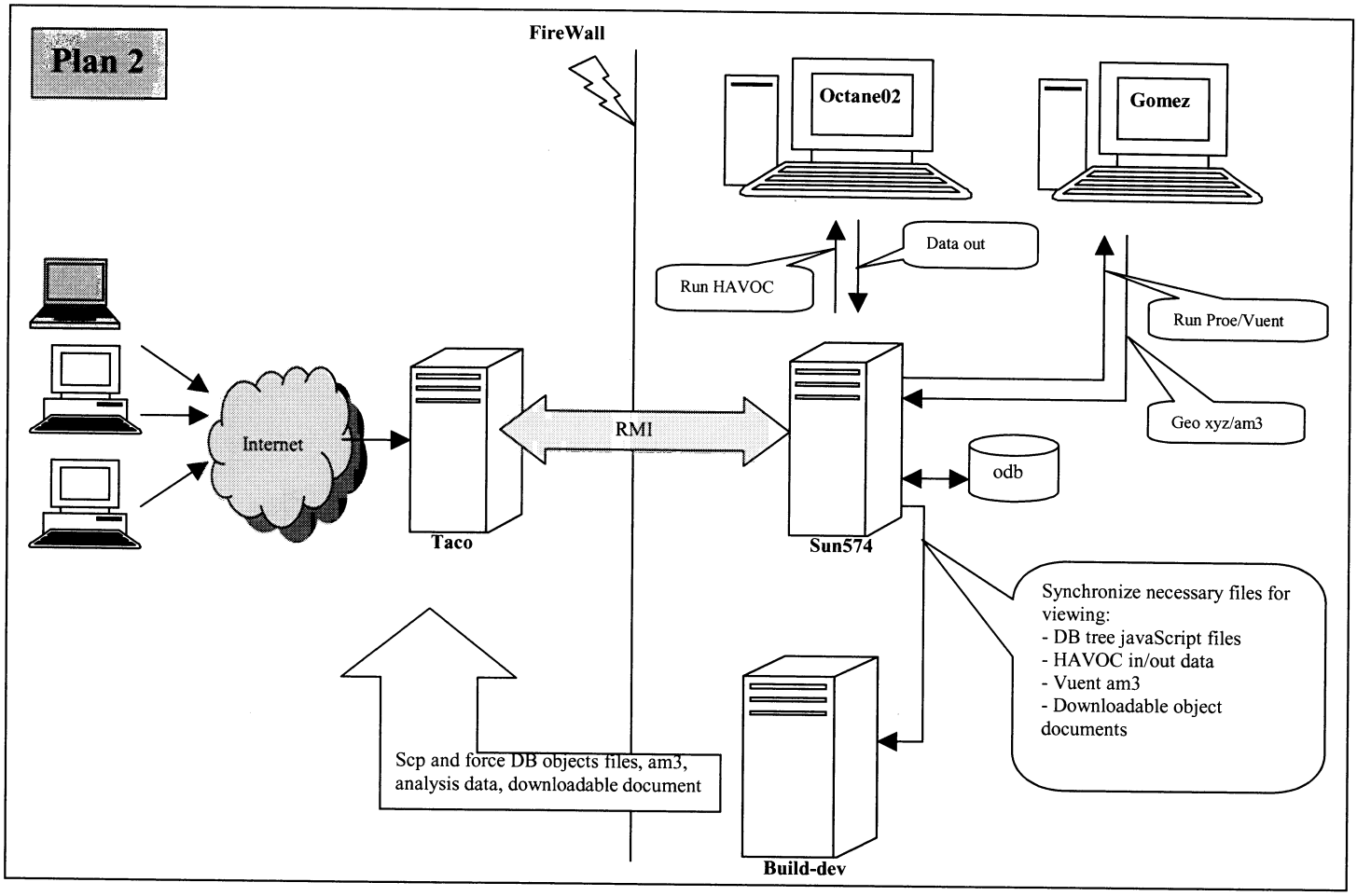

Plan 2
FireWall
Octane02
Gomez
Data out
Run HAVOC
Run Proe/Vuent
Geo xyz/am3
Internet
RMI
odb
Taco
Sun574
Synchronize necessary files for viewing:
- DB tree javaScript files
- HAVOC in/out data
- Vuent am3
- Downloadable object documents
Scp and force DB objects files, am3, analysis data, downloadable document
Build-dev

Figure: 6.6-3

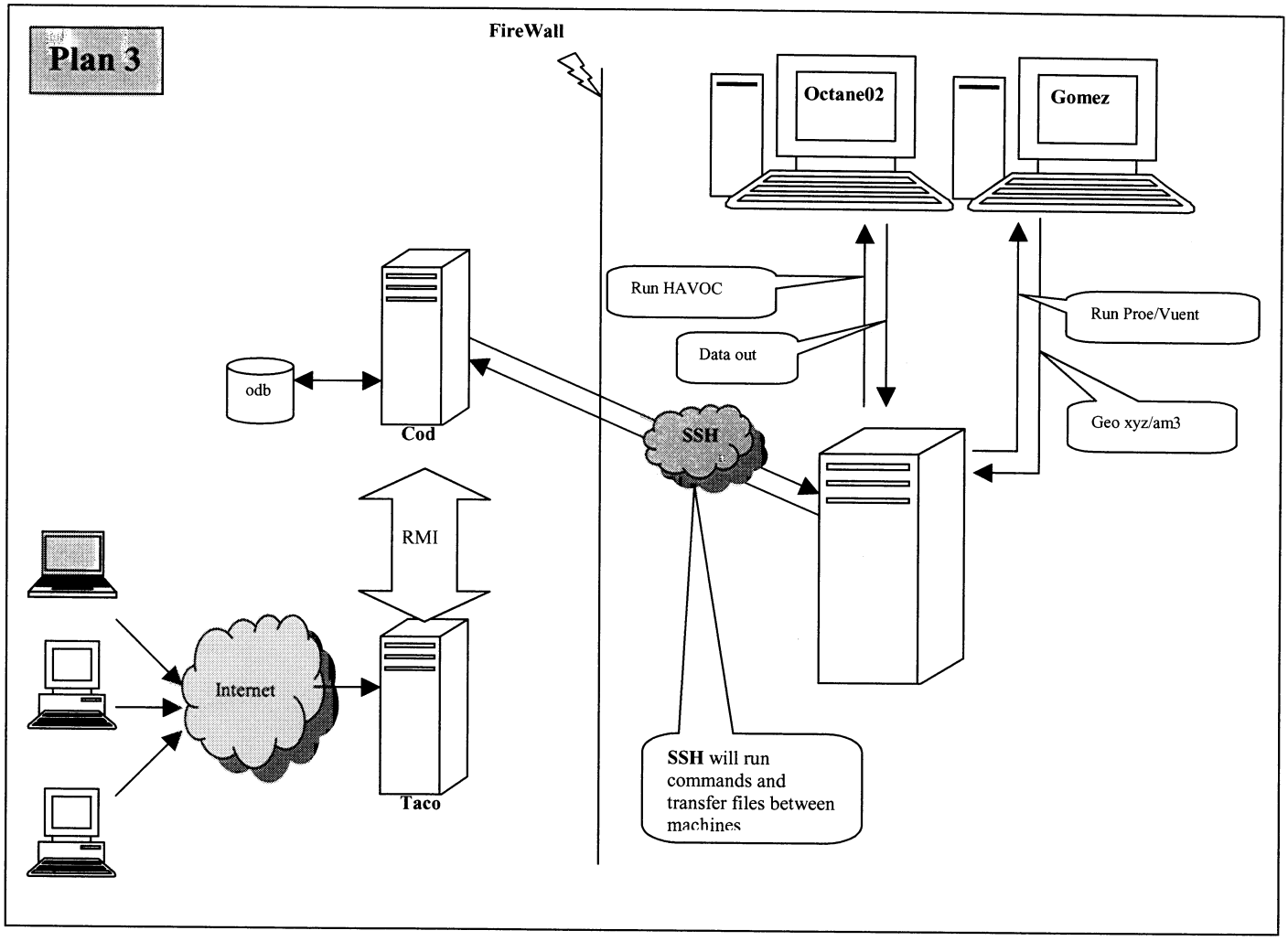
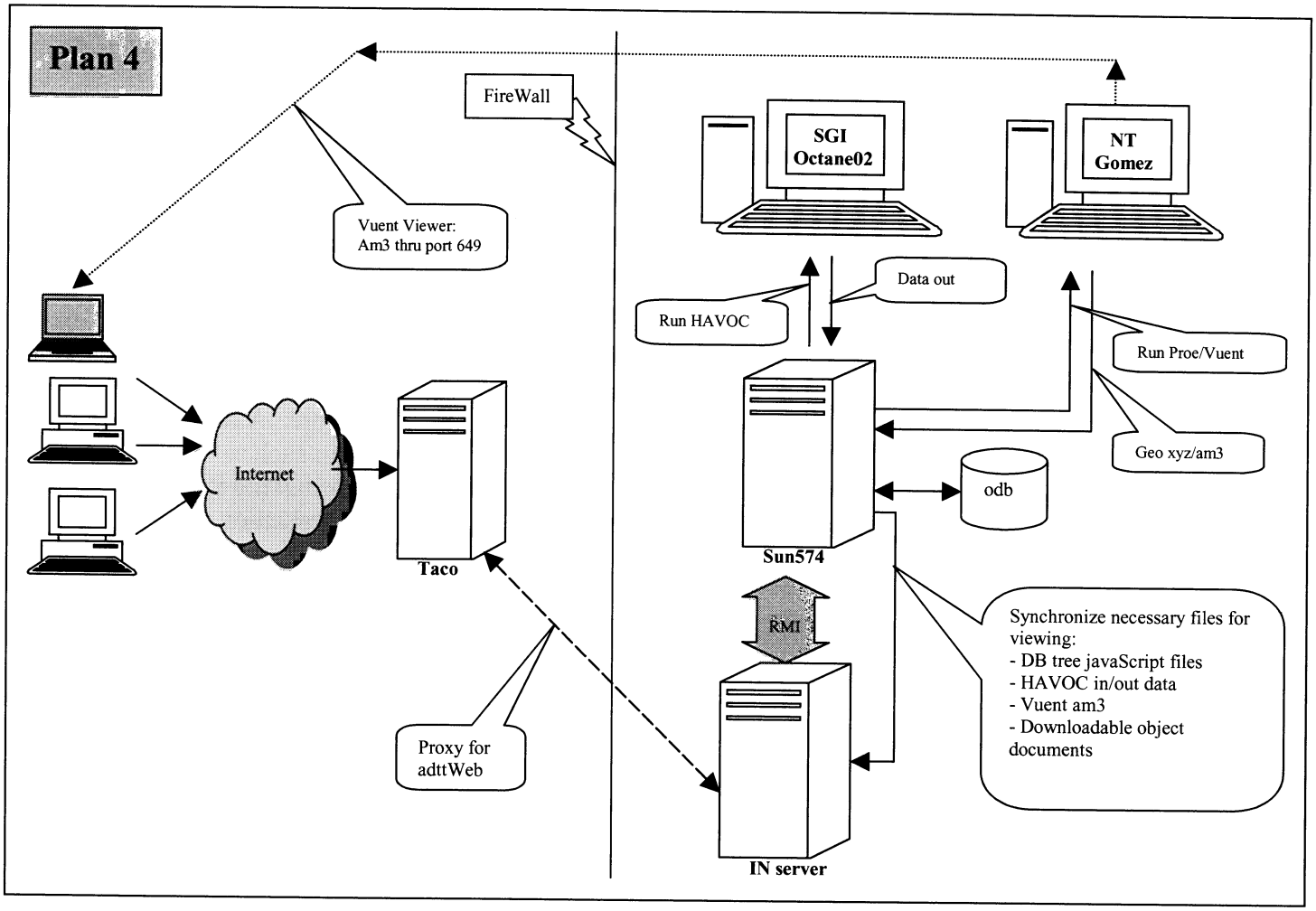

Plan 3
FireWall
Octane02
Gomez
Run HAVOC
Data out
Run Proe/Vuent
Geo xyz/am3
odb
Cod
SSH
RMI
Internet
Taco
SSH will run commands and transfer files between machines

Figure: 6.6-4


Plan 4
FireWall
Vuent Viewer:
Am3 thru port 649
SGI
Octane02
NT
Gomez
Data out
Run HAVOC
Run Proe/Vuent
Geo xyz/am3
Internet
Taco
odb
Sun574
RMI
Synchronize necessary files for viewing:
- DB tree javaScript files
- HAVOC in/out data
- Vuent am3
- Downloadable object documents
Proxy for
adttWeb
IN server

7.0 Summary and Recommendations

The ADTT website has demonstrated the feasibility of bringing together a variety of supporting web based tools to form a cohesive design environment that facilitates teamwork across geographically separated sites. Each of the required elements was evaluated with respect to the tools used for the initial concept, and other tools available. In addition recommendations have been made to apply to future development of this or other similar design environment tools.

Incorporation of the applications required a good deal of work to be done in the areas of developing connecting script programs that work between different computer platforms where the programs were run as well as in the area of security both between internal machines and between off-site logins.

For future planning it is recommended that both Object Store (database) and Vuent (3D Viewing/Collaboration) be replaced by more effective alternatives. A relational database such as Oracle is suggested for replacing Object Store. A replacement for Vuent could consist of something like Actify's Spin with a program such as Netmeeting to support the collaborative element. There are new tools making it to market on a regular basis and some effort should be spent on further evaluation prior to choosing the best tool for production use.

References

1) Summary of Vuent/Envision-i software (Version 5.3) with respect to requirements for ADTT project, November 3, 2000, V. Hawke

2) Summary of Vuent Functionality demonstrated to date with respect to Catia geometry and Vuent/Envision-*i* viewing capabilities, August 25, 2000, V. Hawke

3) ProE – 2 – HAVOC, January 23, 2001, L. Liang

4) Summary of the scripting/programming written to a new modified Catia model using baseline geometry and a new length input from a file, August 1, 2000, V. Hawke

Note: The above references have been included in Appendix B.

Appendix A

Summary of 3rd Party Vendors and 3D Viewing/Collaborative Software

Summary of ADTT Website 3rd Party Software (Currently incorporated and Evaluated)

Tool Type	Company	Application	Version	Comments	Score	Weblink
2D Plotting	**Giga Soft**	**Pro Essentials**	**3.0**	Has decent functionality. Many good plotting features, but missing a few like full chart minor grids. Requires plug-in.	7	http://www.gigasoft.com/
3D Geometry Viewing	Actify	3D View	3.5	Purchased for stand alone use (not web based).	8	http://www.actify.com/3dview.htm
	Actify	SpinFire	1.6	Promising application. Information based on demo at USE conference, evaluation copy not yet available	8	http://www.actify.com/spinfire.htm
	Alibre	Alibre	?	Vault based - not of interest to project.	-	https://www.alibre.com/myprojects/repository.asp
	Centric	CVPD	?	Demo 8/3/00 showed that tool is not all that was advertised. Analysis painting on geometry is only provided via texture mapping, and rest of tool has only some basic functionality.	4	http://www.centricsoftware.com/products/
	Cosmo Software	Cosmo Player	2.1.1	More game oriented.	3	http://web3d.about.com/compute/web3d/gi/dynamic/offsite.htm?site=http://sycamore.inr.net/galaxies.html
	Engineering Animation, Inc	Sense8 Worldup	5.0	Allows individual part path scripting. Does not appear to accept most native CAD formats. Has slightly sub-satisfactory navigation/user interface.	5	http://www.sense8.com
	e-Vis.com	e-vis.com	?	Appears to be vault based, so not of immediate interest to project.	-	https://www2.e-vis.com/project.cgi/app/project/browser/7644/docarchive/64936/view/0?attach=
	PTC	ProductView	5.0	Promising application. Information based on demo at USE conference, evaluation copy not yet available.	8	http://www.ptc.com/products/windchill/collaboration/ds_productview.htm
	SigmaDesignInternational	eZ	1.1	Doesn't appear to be able to bring in CAD formats directly - requires translation to formats such as IGES first.	6	http://www.ezmeeting.com/collab_3Dmodels.html
	Solid-edge	SmartView	8	Primarily a Unigraphics model 3D viewing tool. See viewer summary for details.	4	http://www.solid-edge.com/software/Smartview/smartview.htm
	SolidWorksCorp.	SoilidWorks2000	SP2	Doesn't provide basic needed features. See viewer summary for details.	3	http://www.solidworks.com/html/Products/animator.cfm
	SpatialTechnologyInc.	IntraVision	4.2	Reasonably good tool, but not webased unless used with SpatialTech.Inc. collaboration software.	7	http://www.spatial.com/
	TemplateGraphics SoftwareInc.	3Space Assistant	3	Stereo Views, good navigation, but missing CATIA & ProE file format import capability. Also missing other basic features such as annotation and collaboration.	5	http://www.tgs.com/Assistant/assistant-index.html
	Vuent/iEngineering	**Envision-i**	**5.2**	Provides minimum functionality, non-intuitive interface.	5	http://www.vuent.com/products/
	Wild Tangent	**WebDriver**	**2**	Possible to paint geometry with analytical solutions. User can create 3D viewing environment.	7	http://www.wildtangent.com
Collaboration	GrooveNetworks	Groove	735	Decent collaboration/file transfer/website tour driver/chat tool.	7	http://www.groove.net/
	Microsoft	NetMeeting	2.11	Useful tool, but somewhat slow and dependent on close network links.	6	http://www.microsoft.com/windows/NetMeeting/
	Vuent/iEngineering	**Envision-i**	**5.2**	Collaboration occurs within 3D viewing application. Provides basic chat functionality.	7	
Connectivity	**The Apache Software Foundation**	**Jarkarta-Tomcat**	**3.1**	Provides http server link between NT and Unix. Open source (free)	10	http://www.apache.org
	Xlink Technology Inc.	**Omni-NFS**	**5**	Stably provides directory (file) connection between NT and Unix machines. Some small bugs present.	8	http://www.xlink.com
Database	**OBI**	**ObjectStore**	**6**	Basically works, but is an outdated tool	6	http://www.odi.com/index2.html

Items in bold are currently incorporated into the ADTT Website

Scale 1 worst 10 best

Last Updated: 2/2/2001 14:35

Summary of Viewing Software

Compiled by vmh

File: Birdies//d:/hawke/Other3DGeometry/VeiwerSummary.xls

Last Updated: August 16, 2000

Legend:

Easy The feature works well and has an intuitive interface

Sat. The feature is useable as is (Satisfactory)

Diff. The feature is present but difficult to use and implement (Difficult)

Feature is present.

Upcmg The feature is not present in the current release/trial but is noted as upcoming in the information (Upcoming)

Unkwn The feature may or may not be present, no definitive information could be found

No Feature is not present

													Import											Export					
Company	Application	Version	3D Viewer	Web Based	Modeling	Measurements	Navigation	Collaboration	Annotation/Mark Up	Standard Views	Analysis View		CATIA	Pro/E	AutoCAD	VMRL	IGES	.vda	STEP	STL	SAT	DXF	Unigraphics	.x_t/.x_b	VMRL 1.0	VMRL 2.0	STEP	IGES	SAT	STL	Comments
Actify	3D View	3.5		No[3]	No	Easy	Easy	No	No		No		Easy	Easy		No	Easy	Easy	No	Easy											
Actify	SpinFire	1.5Beta			No		Easy	Upcmg	[8]		No																			
Alibre	Alibre						Sat.				No																			
Centric	CVPD				No	Upcmg	Easy		Upcmg																					Demo 8/3/00
Cosmo Software	Cosmo Player	2.1.1			No	No	Sat.	Unkwn	Unkwn	No	No																			
Engineering Animation, Inc	Sense8 Worldup	5			No	No	Sat.-	[11]	No		No			.slp	.dxf															Part Path Scripting
e-Vis.com	e-vis.com				No	No	Sat.				No																			
PTC	Windchill	5.0									No																			Put In Demo Req.
RealityWave Inc.	VizStream				No	No	Sat.	Unkwn	Unkwn		No																			Streaming
SigmaDesignInternational	eZ	1.1		Unkwn	No	No	Sat.		Unkwn		No																			
Smart Solutions	SmartView/Team/Web																													Have to Call for Demo
Solid-edge	SmartView	8.0		5	6	No	Sat.-	No	No	No	No																			
SolidWorksCorp.	SollidWorks2000	SP2		Unkwn	No	No	Sat.	No	No		No		No	No	No	No	No	No	No	No										
SpatialTechnologyInc.	IntraVision	4.2		10			Sat.-	10			No		Easy	Easy																
TemplateGraphicsSoftwareInc.	3Space Assistant	3			No	No	Easy	No	No	No	No		No	No			[4]	[4]	[4]	[4]										Stereo Views
Vuent/iEngineering	Envision-i	5.2			No	Diff.	Diff.	Sat.	Sat.[2]	No	No		[4]	[4]							[9]									

See Notes next page

Notes

2 Some problems with geometry resizing with window re-size when viewing and re-loading an annotation

3 See SpinFire application from same company

4 Using a translator application

5 This stand-alone viewer can also be run with an ActiveX document container such as Internet Explorer or Netscape.

6 Modeling available in other software from the same company

7 Top/Sides/Front/Back/Bottom/Iso

8 Text only, requires (Ctrl+Alt+Print Screen) and paste to another document to save.

9 May allow other import files with translators (Need to check)

10 Web based collaborative software available from the same company

11 Networking through browser

Appendix B

Supplemental Reports

Summary of Vuent Functionality demonstrated to date with respect to Catia geometry and Vuent/Envision-*i* viewing capabilities

Document: Birdies//D:/hawke/Vuent_Stuff/Demonstrated.doc

Author: Veronica Hawke

Last Updated: August 25, 2000

Table of Contents

Split Baseline/Modified Geometry

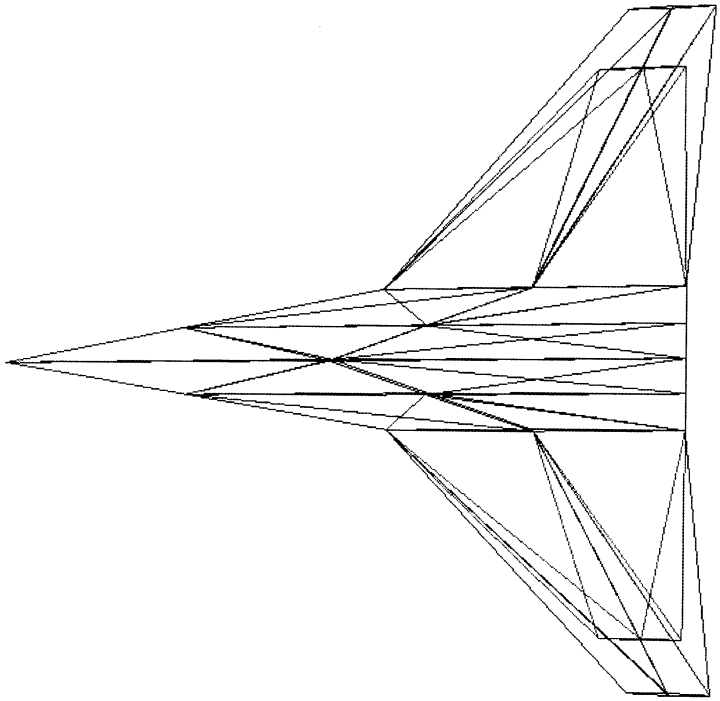
Steps:

1. Two separate Catia models created each with both halves created as separate solids.
2. Catia models run through Vuent Catia Translator independently.
3. Resulting .3dx files brought together in one Workspace/Project in Vuent Optimizer
4. Opposite sides hidden in Vuent viewing session via the Parts Manager.
5. View saved using 'Save' button and selecting jpeg format.

Discussion:

It is relatively easy to create half models in Catia and mirror them across any defined plane. Both baseline and modified geometry should share a common reference system to ensure the comparison turns out as desired. Note that the creation of the modified geometry was done by manually moving the extended defining points and re-creating the revolved surface based on the new outline defined by those points.

Wireframe Baseline/Modified Geometry



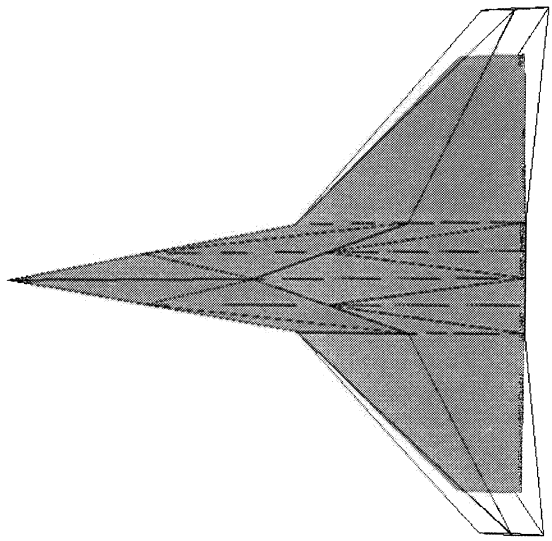
Steps:

1. Two separate Catia models created using lines based on points to create surfaces.
2. Catia models run through Vuent Catia Translator independently.
3. Resulting .3dx files brought together in one Workspace/Project in Vuent Optimizer.
4. Parts set to different contrasting colors.
5. Wireframe viewing selected for all parts.
6. View saved using 'Save' button and selecting jpeg format.

Discussion:

Note that the default triangle wireframe barely provides enough definition for overlay comparison purposes.

Solid Baseline/Wireframe Modified Geometry



Steps:

1. Two separate Catia models created using lines based on points to create surfaces.
2. Catia models run through Vuent Catia Translator independently.
3. Resulting .3dx files brought together in one Workspace/Project in Vuent Optimizer.
4. Baseline part set to contrasting color.
5. 'Pick' color set to contrasting color via Preferences Menu (right mouse).
6. Modified part selected with Assembly Manager and using the right mouse 'Select'.
7. View saved using 'Save' button and selecting jpeg format.

Discussion:

This was set up, using the selected state viewing characteristics to provide the comparison against the unselected state of the baseline geometry. Fortunately the viewer allows the navigational tools to work in their regular modes while the geometry is in this state. Again work with the color choices may produce better results. The Path Manager works on the geometry in whatever state it is in including this one.

Programmed Viewing Path

Discussion:

Through the use of an uuencoded text file a basic viewing path has been set up to run the geometry through the standard views. This is accomplished by opening the modified path in the Path Manager. A general file has been set-up that should work with any geometry approximately the size of the fly-back-booster design. The file currently resides in Birdies//d:\hawke\Vuent_Stuff\booster_view_in.path.

Summary of Vuent/Envision-i software (Version 5.3) with respect to requirements for ADTT project.

Document: Birdies//D:/hawke/Vuent_Stuff/Evaluation5_3.doc

Author: Veronica Hawke

Last Updated: 3 November 2000

Table of Contents

Introduction

Vuent's (now iEngineering) Envision-i software is being evaluated as a component of the ADTT project which will provide a collaborative database and design working environment for the next generation of aerospace vehicles. Vuent's software has been chosen for evaluation because of its ability to provide a collaborative environment with an efficient method of presenting geometric changes to the proposed models as well as it's ability to provide a way to document collaborative sessions through saved annotations and typed comments.

This summary will address the list of items and features in Vuent's Envision-i software which may require further refinement or alterations to make the product produce the desired results. This document is based on the evaluation of Version 5.3.

Ability to Save Annotation Image to a Picture File

It is not obvious at this point how the user goes about capturing any of the annotated images to the clipboard or to a picture file such as jpeg, bmp, gif, etc... While in the annotation manager all of the standard tool bars are hidden including the 'copy to clipboard' button. While this may not be a critical item in general, it would allow a record of the size of window at which the annotation was originally recorded. It should be possible to capture a view with a programmed API button.

Navigation Tools

The navigation tools provided in the application interface are tricky to use and tend to be counter-intuitive. The basic idea is to provide the user with rotation, translation and zoom capabilities within one mode of operation (keyboard, mouse, or toolbar), with only minimal requirements to use an action outside of the desired mode. For the demonstration it is likely these tools will be either partially turned off and/or rearranged to provide the user with a minimally useable set that is less likely to lead to confusion while manipulating the view. A proposed arrangement of the tools is shown in Figure 1 below.

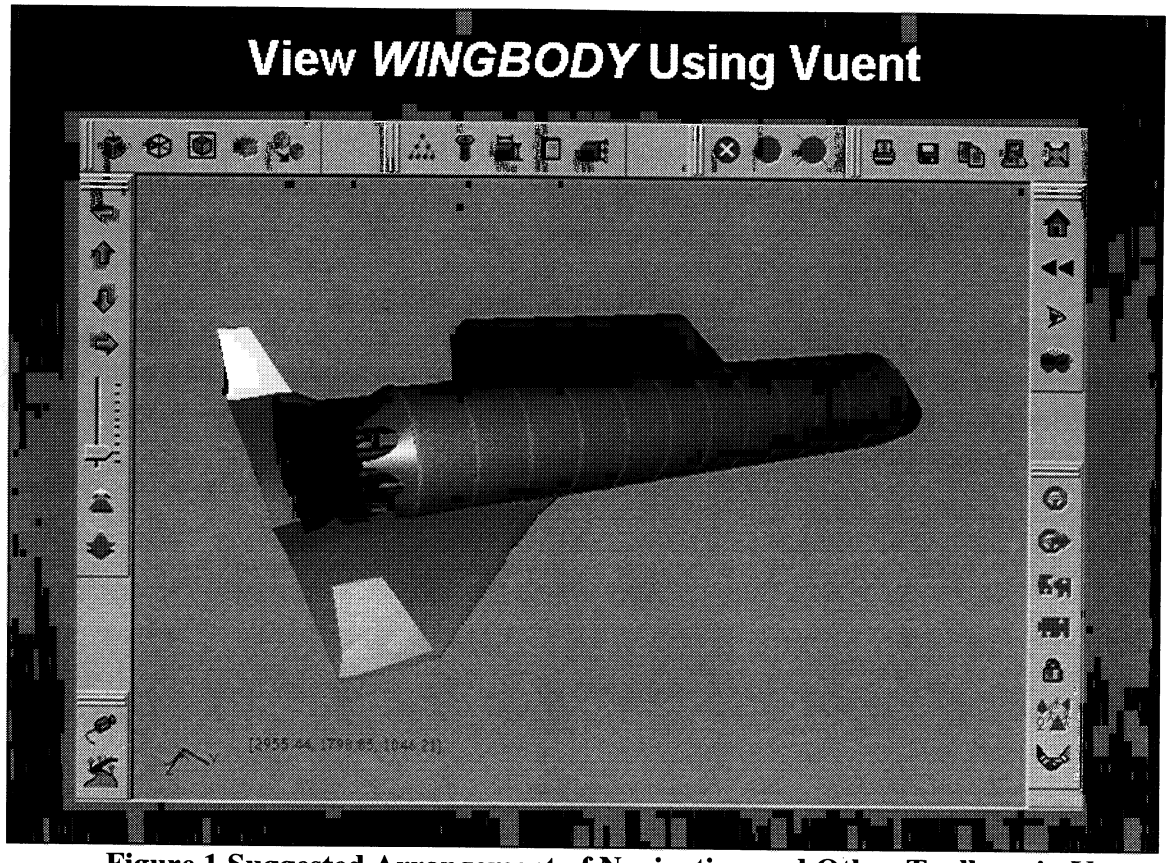


Figure 1 Suggested Arrangement of Navigation and Other Toolbars in Vuent

Non-Perspective Viewing

The inability to toggle the perspective view on and off detracts from the functionality of the viewer. Non-perspective views assist in geometric evaluations of things such as parallel lines and degree of rotation. The perspective view leads to more difficulty in assuring the accuracy of the measurement manger, especially since the measurement is limited to selection of end points in a frozen view of the model's current position.

Standard Views

It would be helpful for a set of standard views to be available automatically. Vuent's two (front and side views) small side pop-up windows are only marginally helpful due to their size and inability to set the main window to a given view. It is possible to set up a set of shared views for each geometry model, but this requires extra time to provide a functionally that is commonly included in many viewing applications. Also the user's set views are all in perspective mode, reducing their usefulness.

User Color for Collaboration

While in a collaboration session each user is assigned a color by the program. It is not possible to change this color setting within the collaboration session or prior to the session in the preferences menu. This leads to problems when a user's background color happens to coincide with the assigned color. For example the first color assigned is white, which is a common choice for a background color. If the collaboration color matches the user's background color, none of the annotations done by that user are visible.

Wireframe Selection During Collaboration

If the driver of a collaborative session selects the wireframe viewing option for the model it shows up on only the driver's screen and not the other client screens. If individual parts are selected by the driver and highlighted via the wireframe the client views appropriately show the selection in wireframe.

Assembly/Parts Managers

The windows for the assembly and parts managers are separate entities in Vuent. Though this may be useful in some context it is an arbitrary division when basing the hierarchy on CAD geometries. It would be smoother and simpler for our purposes to have one window open with the geometry organized in the same way as the CAD geometry.

Annotation Image

Below are two pictures showing the effect of re-sizing the window on a saved annotation image. Figure 2 is close to the size at which the original annotation was saved. Figure 3 is of the same image in a different size window.

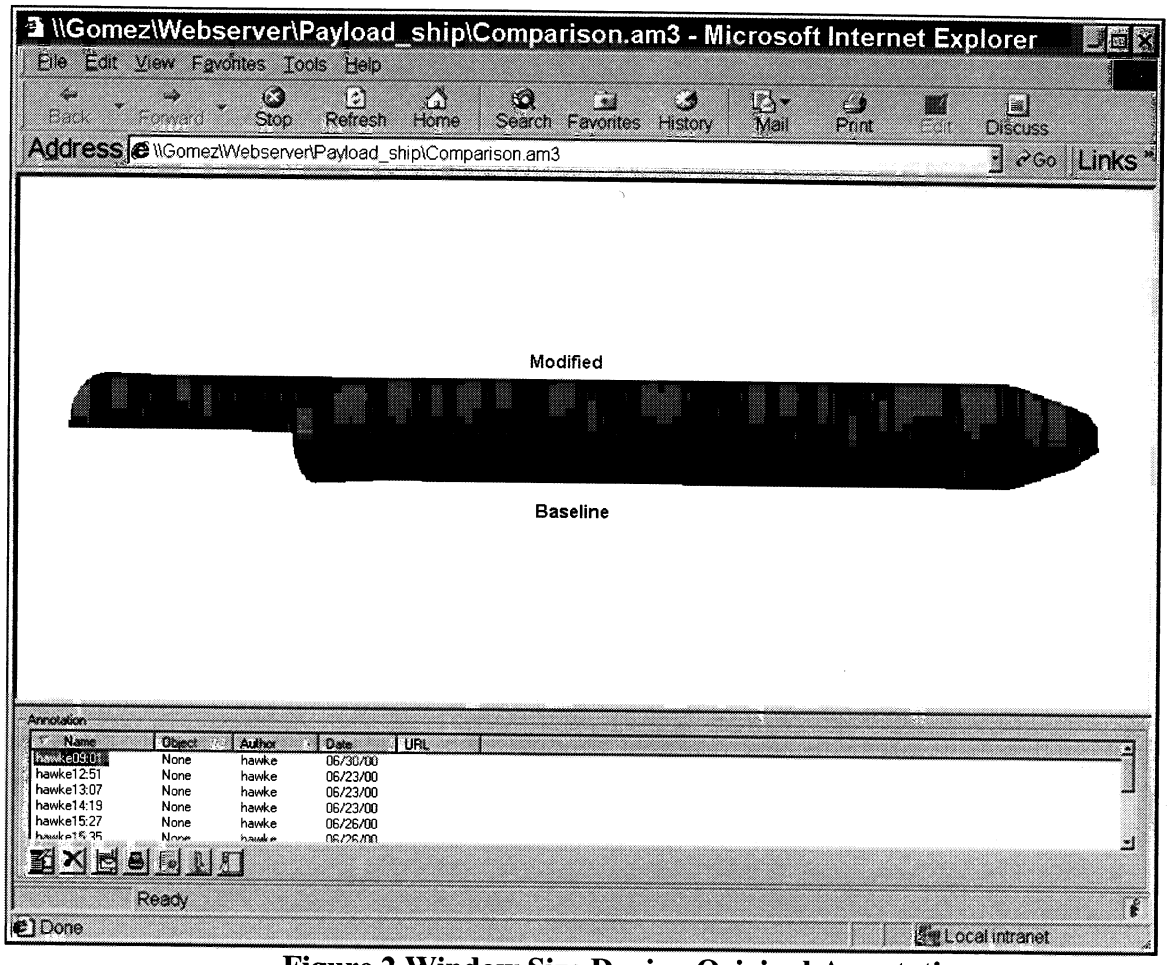


Figure 2 Window Size During Original Annotation

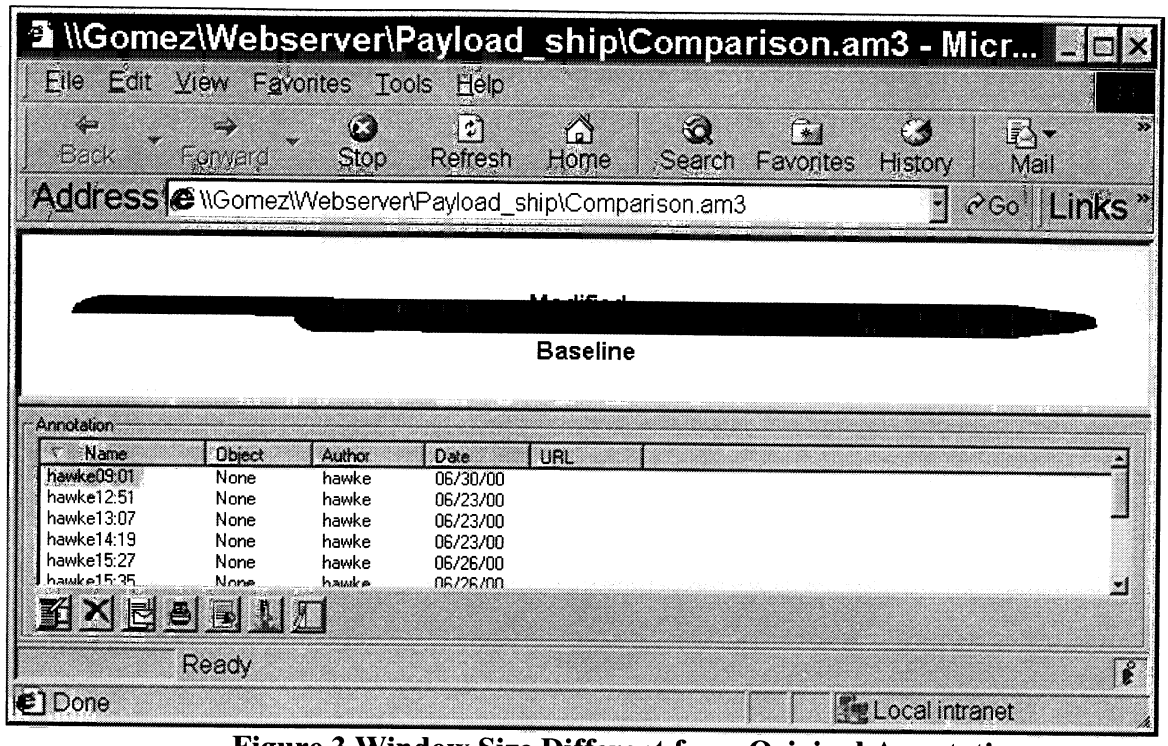


Figure 3 Window Size Different from Original Annotation

The window size affects the view of the geometry. The concern is that the geometry appears modified when no changes have yet been made. There is no standard window size and it is likely that every user will have different window sizes. The geometry needs to be presented in it's original aspect ratio to be accurately represented. This is a serious concern however no fix has been offered in time for the upcoming demonstration.

Path Manager

In the **Advanced** view of the path manager, the representation of the line that is followed is in white, causing it to all but disappear when the background is set to white. See Figure 4 for illustration. This is a very minor concern as at least the direct lines between the views are shown in gray. It simply makes reviewing a path more difficult. One other comment about the Path Manager is that the fixed size of the viewing window given to work in and the inability to zoom out further make modifying a path difficult as well, unless the text form is used to managed the desired views.

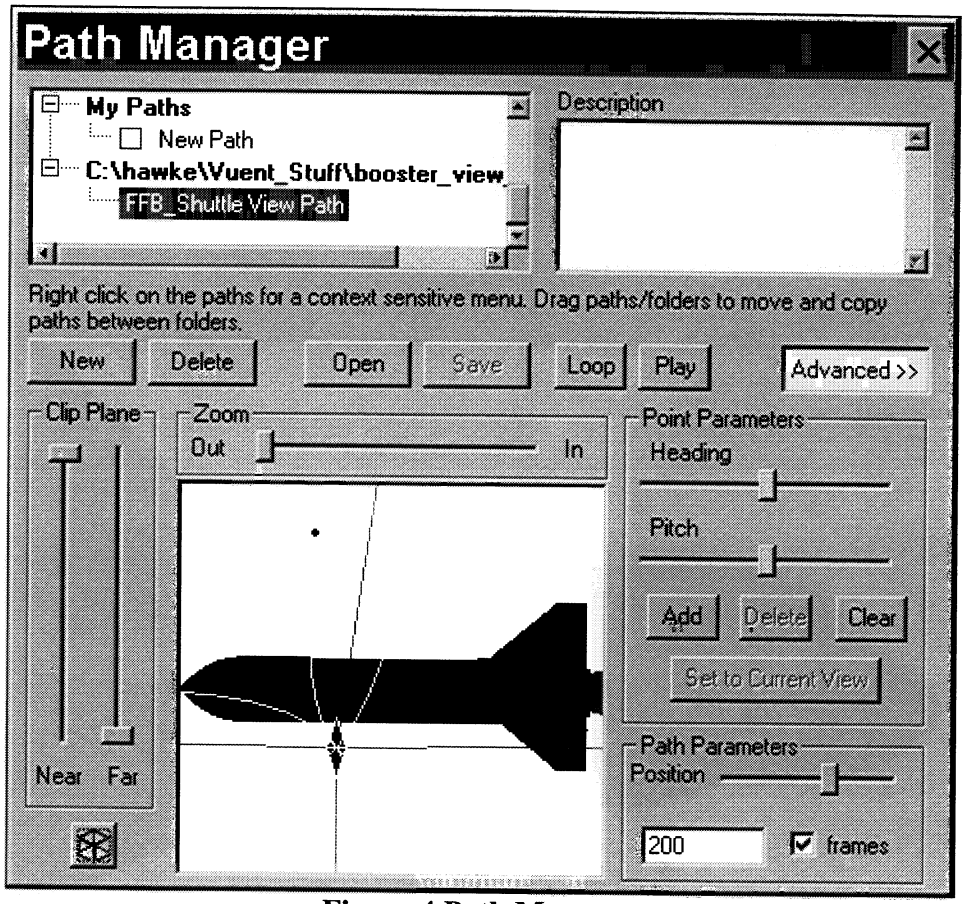


Figure 4 Path Manager

Measurement Manager

The measurement manager is only of marginal use. Though it measures distance or angles from triangle to triangle it lacks the ability to label a point with XYZ coordinates. The values for XYZ in the lower left hand corner of Figure 5 happen to correlate with the last arrow from the last measurement and can not be copied for placement on the view. In addition the interface doesn't always work as expected. Occasionally it is difficult to select the first point where desired, resulting in a '0' distance.

A more critical problem with using the measurement manager is the inability to control or specify exactly which vertex of a triangle is being used for a measurement. Figure 6 shows an example of how this can cause seriously misleading results. Overall this limits the user's access to model details.

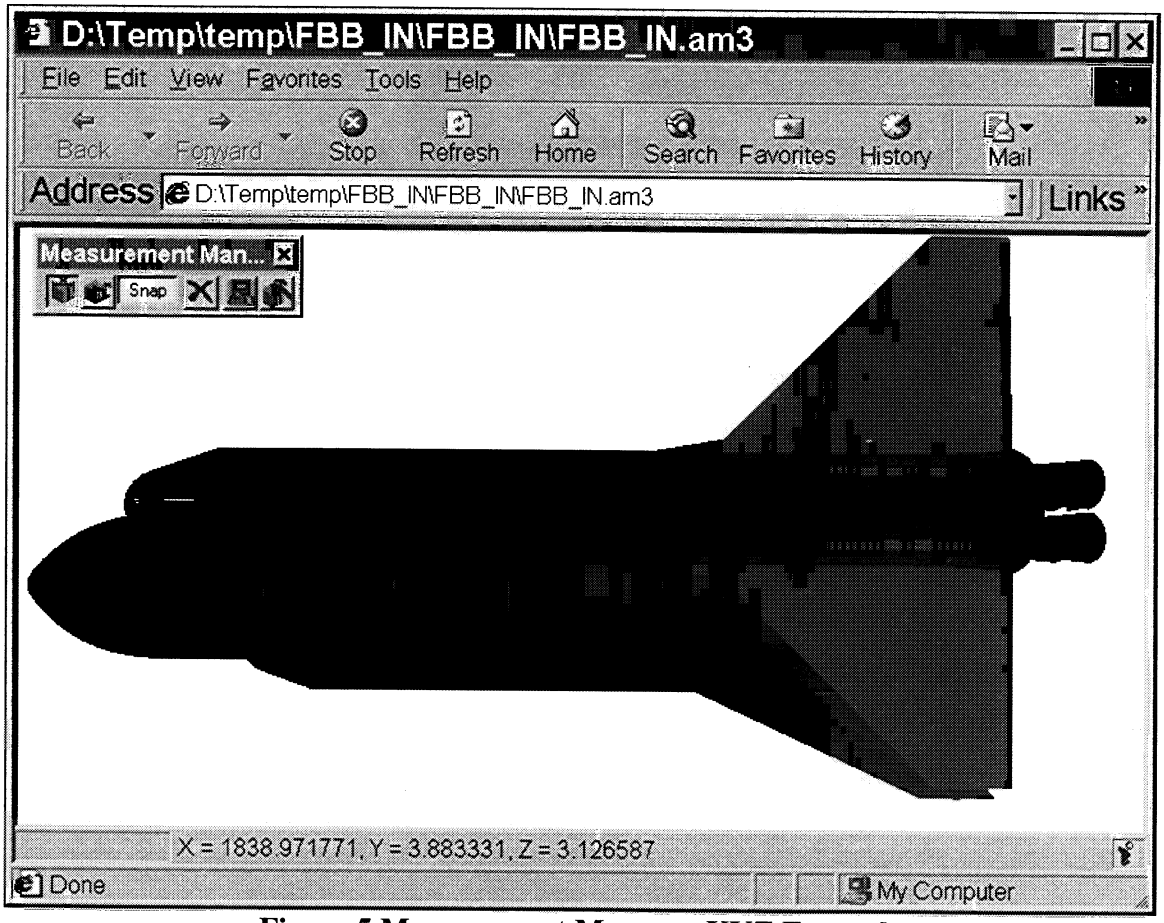


Figure 5 Measurement Manager XYZ Example

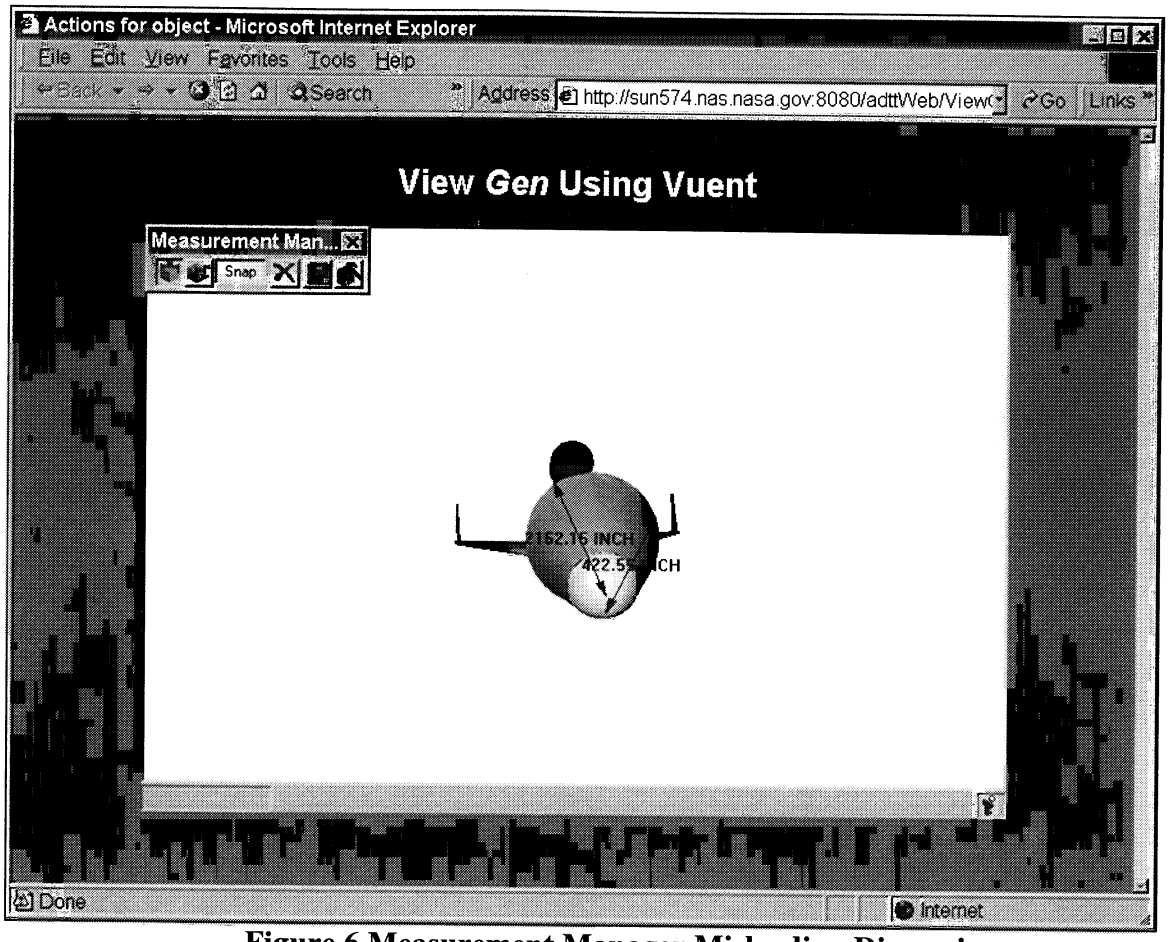


Figure 6 Measurement Manager Misleading Dimensions

ProE – 2 – HAVOC

Prepared by:	Lawrence (Fang H.) Liang Eloret Corp.

Revision Sheet			
Section	**Rev.**	**Reason For Revision**	**Date**
All	--	Initial Release. LFH	01/23/01

DEFINITIONS

Term	Definition
ADTT	**Advanced Design Technologies Testbed**
ProE	**Pro Engineer, CAD package from Parametric Tech. Corp.**
IGES	**Initial Graphics Exchange Specification**
OML	**Outer Mold Line**
HAVOC	**Hypersonic Aerospace Vehicle Optimization Code**

Table of Contents

List of Figures

Appendix

1 Purpose/ Background

The purpose of this document is to document the procedures to process geometry data from Pro-Engineer (ProE) to a HAVOC readable file format. This process was done in supporting the ADTT effort; in which, the ProE geometries were parameterized to accept geometry changes made from the ADTT web site. This updated geometry is then processed and used to run HAVOC.

A total of 4 different 2nd Gen. geometry were constructed for this project: Lifting-body, Bimese, Wing-Body, and SST w/ Fly Back Boosters. Most of the parameterization works were done solely on the Wing Body geometry.

2 HAVOC Input Requirements

HAVOC separates the vehicles in terms of fuselages, and lifting surface components. The OML of these components is defined as a series of points on the OML, which is written into its own *.xyz ASCII files.

For the fuselage, the .xyz files organize these points as a set of cross section points, distributed all along the fuselage. See Figure 1 for a sample of the fuselage file with 4 cross-sections distributed along the fuselage (X-axis), and with 5 points distributed along the cross-section outline. The point distribution at each cross section starts at the at the maximum Y location and continues on in a clockwise fashion.

```
 04      (Number of Cross Sections)
 05      (Number of points/Cross-Section)
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.0000000000000000
0.0000000000000000 0.0000000000000000 0.0000000000000000
 05
0.2504800739989150 0.0031145999673754 0.0000677034258842
0.2504800739989150 0.0030554300174117 0.0006767027080059
0.2504800739989150 0.0028787299524993 -0.0011257976293564
0.2504800739989150 0.0030554300174117 -0.0005411989986897
0.2504800739989150 0.0031145999673754 0.0000677034258842
 05
0.6018917600391433 0.0080752000212669 0.0001756027340889
0.6018917600391433 0.0079217301681638 0.0017544999718666
0.6018917600391433 0.0074636498466134 0.0032702013850212
0.6018917600391433 0.0079217301681638 -0.0014031976461411
0.6018917600391433 0.0080752000212669 0.0001756027340889
 05
1.0000000000000000 0.0136903002858162 0.0002977028489113
1.0000000000000000 0.0134300999343395 0.0029744021594524
1.0000000000000000 0.0126534998416901 0.0055440030992031
1.0000000000000000 0.0134300999343395 -0.0023790001869202
1.0000000000000000 0.0136903002858162 0.0002977028489113
 -999
-99999
```

All ZEROs for the Nose

First Pt. = Last Pt.

X=1.0 at last C-section

Figure 1 , Sample Fuselage *.xyz File

For the lifting surfaces, the file is formatted in a similar fashion, but omitting the cross-section number, see Figure 2 for a sample of the ASCII file. The points starts at the leading edge, and then progress toward the trailing edge on the upper surface, and then comes back to the leading edge via the lower surface.

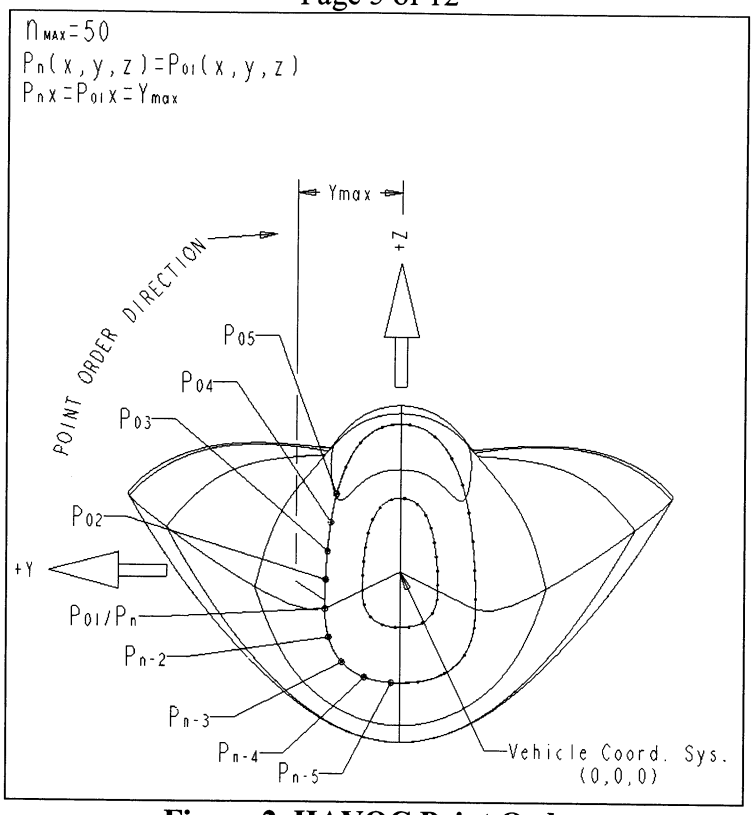


Figure 2, HAVOC Point Order

3 ProE Geometry Generation

3.1 Assembly Structure

All the geometries in the Wing-Body vehicle are assembled in ProE in a fashion shown in Figure 3. This assembly structure information is then exported as a bill-of-material (BOM) ASCII file to be read by the ADTT database (Figure 4).

3.2 Scale

HAVOC requires the fuselage length to be equal to 1, and so all other components of its geometry need to be scaled accordingly (to the same scale). Construction of the ProE geometry is built on actual size of the vehicle. Scaling of the exported points are handled at post possessing, after the OML points are exported onto a IGES file.

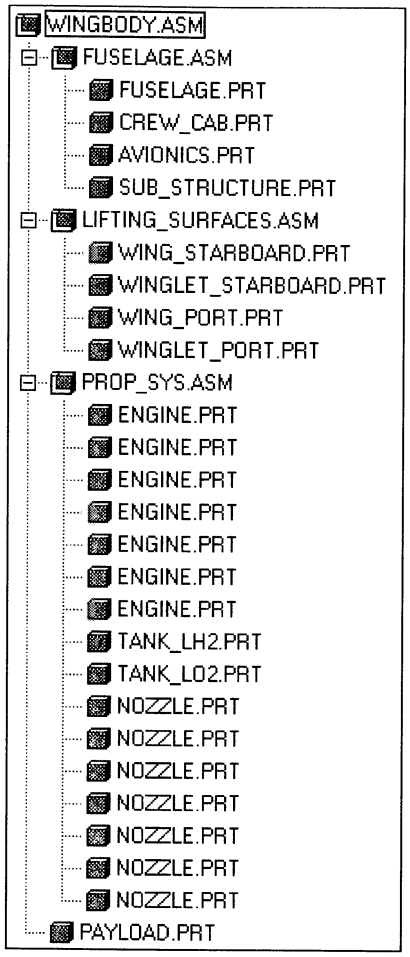


Figure 3, Wing-Body Assembly Tree

```
Assembly WINGBODY contains:
   1 Sub-Assembly  FUSELAGE
   1 Sub-Assembly  LIFTING_SURFACES
   1 Sub-Assembly  PROP_SYS
   1       Part  PAYLOAD

Sub-Assembly FUSELAGE contains:
   1       Part  FUSELAGE
   1       Part  CREW_CAB
   1       Part  AVIONICS
   1       Part  SUB_STRUCTURE

Sub-Assembly LIFTING_SURFACES contains:
   1       Part  WING_STARBOARD
   1       Part  WINGLET_STARBOARD
   1       Part  WING_PORT
   1       Part  WINGLET_PORT

Sub-Assembly PROP_SYS contains:
   7       Part  ENGINE
   1       Part  TANK_LH2
   1       Part  TANK_LO2
   7       Part  NOZZLE

Summary of parts for assembly WINGBODY:
   1       Part  FUSELAGE
   1       Part  CREW_CAB
   1       Part  AVIONICS
   1       Part  SUB_STRUCTURE
   1       Part  WING_STARBOARD
   1       Part  WINGLET_STARBOARD
   1       Part  WING_PORT
   1       Part  WINGLET_PORT
   7       Part  ENGINE
   1       Part  TANK_LH2
   1       Part  TANK_LO2
   7       Part  NOZZLE
   1       Part  PAYLOAD
```

Figure 4, Wing-Body BOM Output File

3.3 Coordinate System

HAVOC locates the wings relative to the nose of the fuselage. The nose of the fuselage is set to be at (0,0,0), and the base/tail of the vehicle is at (1, y, z). The first point of the wing components is set to be at the LE of the first wing cross-section. See Figure 5.

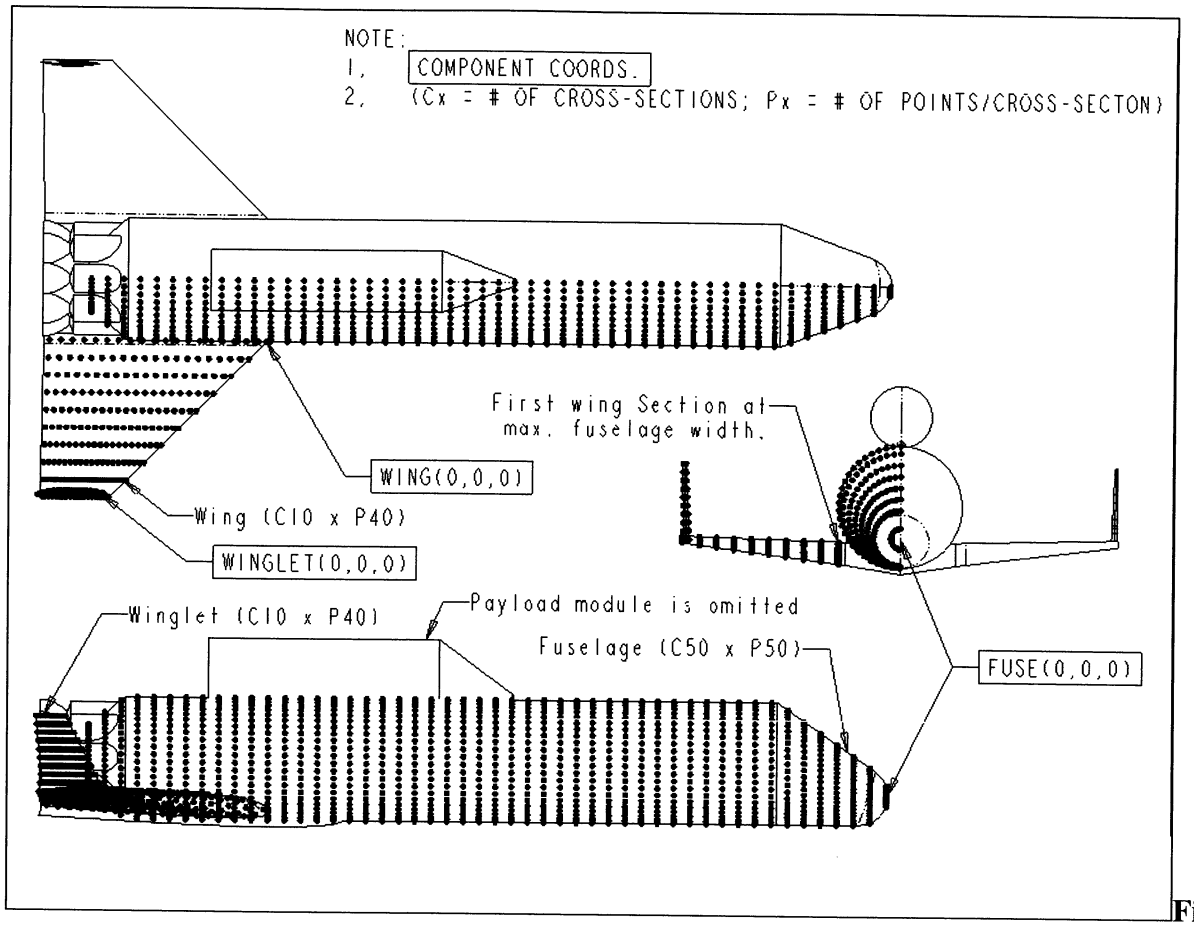


Figure 5, Wing-Body Geometry

All ProE geometry components are assembled by coordinate systems, so as to facilitate parameterization of the vehicle. In each of the components, a global coordinate system is created, and the whole vehicle is then assembled with this coordinate system. Parameters used to create this coordinate system are then are exported along with other parameters to be processed, and used by HAVOC to locate these components.

3.4 Fuselage

There are many ways to construct the fuselage inside ProE, and several methods were experimented with during the course of the ADTT development.

Generating these points on a solid body or an enclosed surface (with both side of fuselage present) prove to be problematic. Points have a tendancy to bounce up in some cases, and failed to circuscribe the whole cross-section. In some cases, it is impossible to control the start location of these points along the curve.

One of the more successful ways of creating a geometry for the purpose of generating *.xyz files, was to first construct a surface definition of the half geometry, and then create a datum curves on that surface which is parallel to the YZ plane. After the datum curve is created, points are then created along this curve. This method does require more post processing work to be done after the points are exported, but it proved to be more reliable than the other schemes. The following steps, describe the process of generating these points in ProE:

1. Set up Coordinate Sys. and a set of datum planes(XY, YZ, and XZ)

2. Construct the half fuselage geometry on the +Y side of the coord. Sys.
3. Create a datum plane offset from the XZ plane, intersecting the OML just behind the nose.
4. Create a datum curve by intersecting the datum plane with the halved-OML
5. Create a datum point on the curve, by length ratio.
6. Set the ratio to 0, so the point rest on the bottom end of the curve(the most -Z point)
7. Pattern the points along that curve, spacing them evenly with the last point resting at the upper end of that curve.
8. Group the datum plane, the datum curve, and the patterned datum points into a local group.
9. Pattern the group along the fuselage.
10. Mirror the other half of the vehicle if needed.

NOTE 1: When patterning the points along the cross-section line Step 7, set it to be 26 (The maximum number HAVOC is capable of handling). If a lower relsulotion points cloud is needed, reduce this number ONLY after the Groups are Patterned (Step 7). The number of points along the line can not be increased after the Group is patterned. Doing so,will cause the model to fail to regenerate.

3.5 Lifting Surfaces

Lifting Surface OMLK points are constructed in a similar fashion as the fuselage, with the exception that the first point starts at the leading edge. For this exercise, the wing surface of the Wing-Body geometry is constructed of a single surface, combining the top and bottom surface. While this presented little problem in locating the starting point for this particular wing. For future work, it is recommended to separate the top and bottom wing surfaces, so to guarantee proper point order.

3.6 Parameterization and Pro/Program

Each and every component of the Wing-Body is parameterized in some manner. The Appendix contains three drawings of the WB geometry with the driving parameters identified. These drawings and others are posted on the ADTT site for the users as a reference. All parameters are set up to be able to drive the geometry changes from a text file input. This was done with the use of Pro/Program and trail file. Figure 6 shows a parameter file used for the WB fuselage. These parameters are extracted from the ADTT database, and reformatted appropriately for Pro/Program as input.

FUSE_LENGTH	= 205.000000
NOSE_LENGTH	= 28.905000
FUSE_DIA	= 32.000000
NOSE_RADIUS	= 7.720000
NOSE_HEIGHT	= 7.600000
AFT_DIA	= 16.85
TRANS_X	= 0.0
TRANS_Y	= 0.000000
TRANS_Z	= 8.2800000

Figure 6, Sample Parameter File

3.7 Point Export and Trail File

All the generated points on the vehicles are exported in the form of an IGES file. Three files are generated, one for each OML part. The payload module was omitted, because HAVOC can only handle a single fuselage geometry.

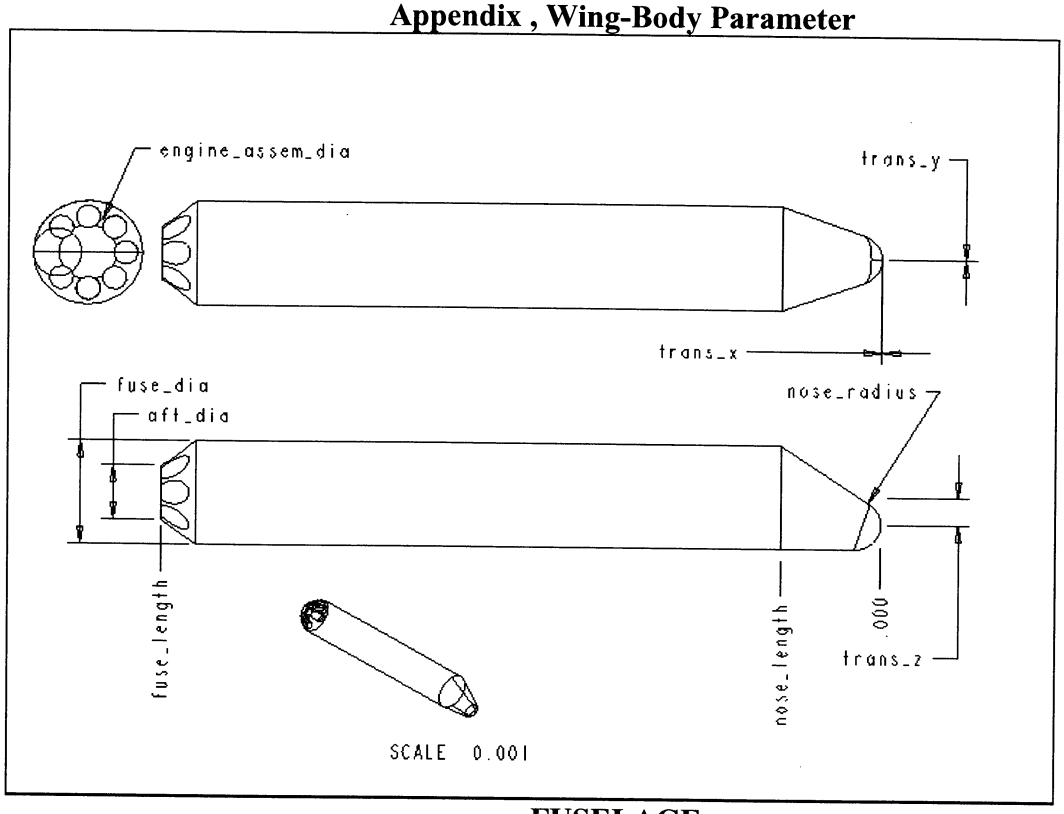
The processes of generating and updating the geometry from the new parameters are all handled with a pre-recorded trail file. The trail file will read-in the new value from the parameter file and regenerate the geometry for each one of the components. The trail file then proceeds to generate the IGES files of the fuselage, wing, and winglet. During the IGES export operation, care was taken to filter out other geometry entities from the IGES files by using layers control.

The IGES files are then post processed into the *.xyz format for HAVOC. A Perl script is used, first to extract these points from the IGES file, then it goes through mirroring those points, and finally reordering and scaling them accordingly.

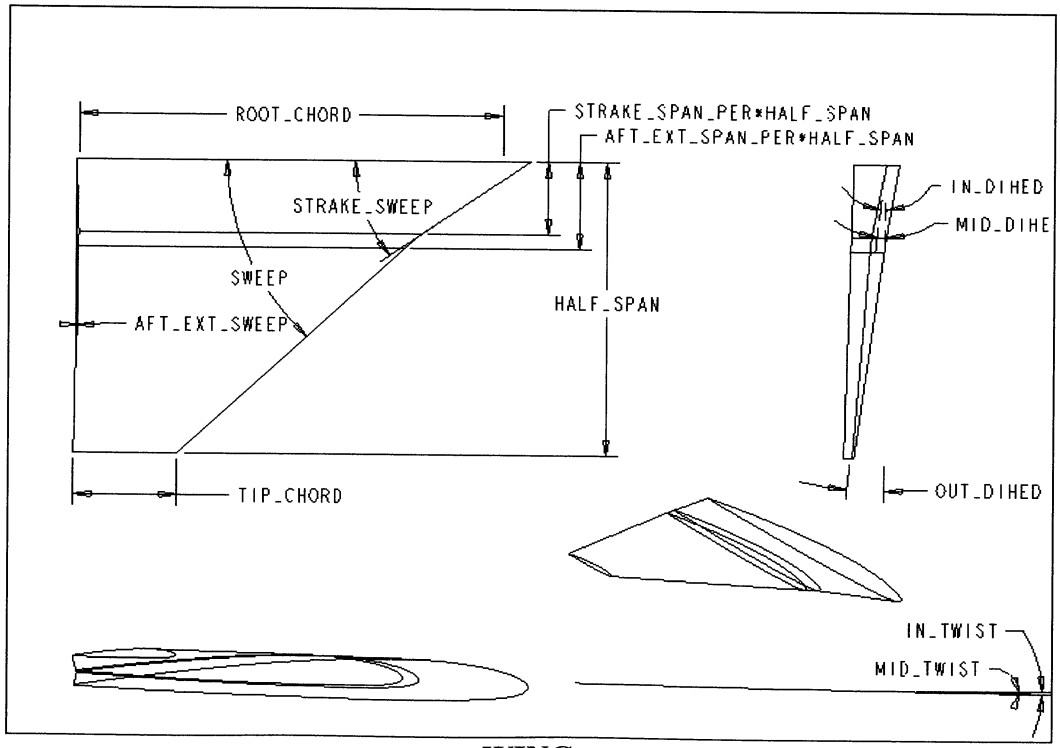
4 Results

The geometry generated from this process was able to produce a valid HAVOC xyz geometry, and was able to perform the geometry updated as required.

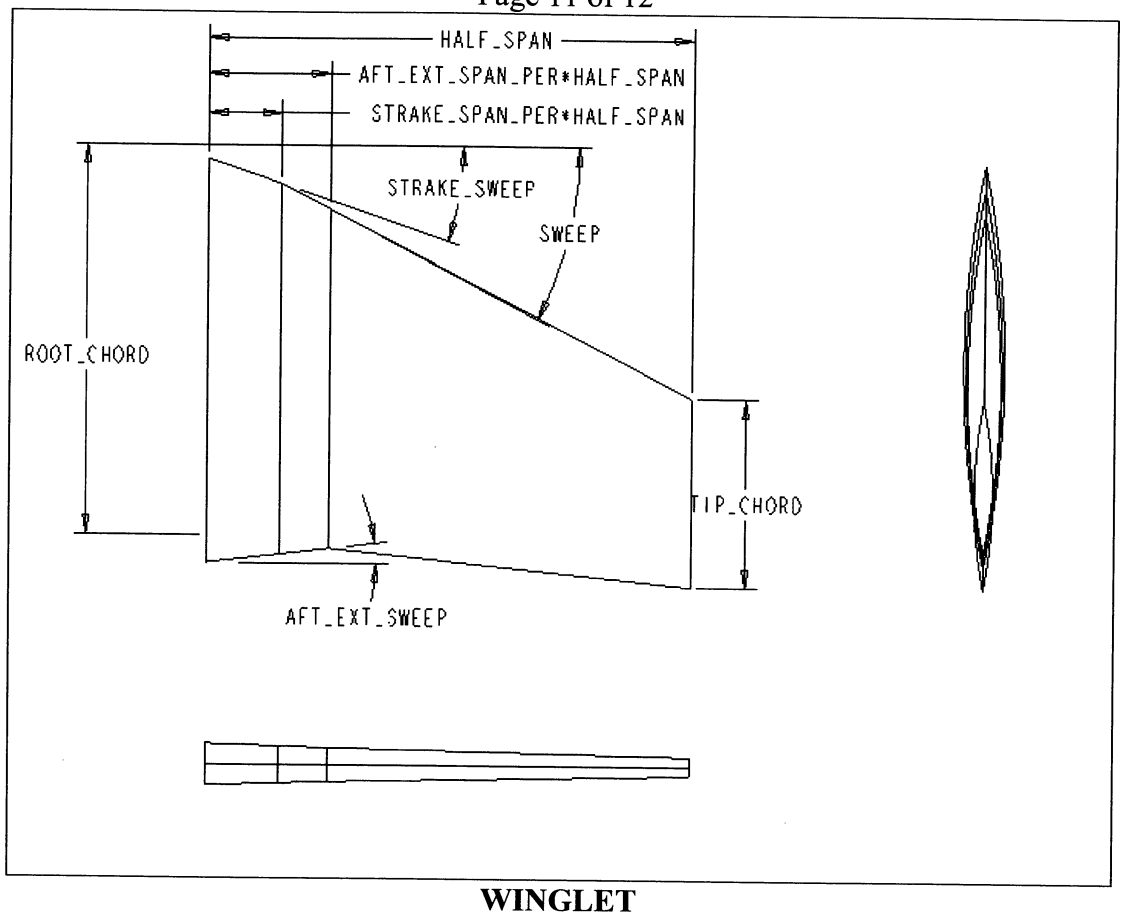
Appendix , Wing-Body Parameter



FUSELAGE



WING


HALF_SPAN
AFT_EXT_SPAN_PER*HALF_SPAN
STRAKE_SPAN_PER*HALF_SPAN
STRAKE_SWEEP
SWEEP
ROOT_CHORD
TIP_CHORD
AFT_EXT_SWEEP

WINGLET

Summary of the scripting/programming written to a new modified Catia model using baseline geometry and a new length input from a file.

Document: Birdies//d:/hawke/Reports/CatiaGeometryScript.doc

Author: Veronica Hawke

Last Updated: August 1, 2000

Table of Contents

Overview

The purpose of this task was to demonstrate the capability of changing a geometry parameter in a Catia model and creating the updated model, using a scripted routine. The program documented by this report is not intended as a final or production code, only as a 'proof-of-concept' demonstration of functionality. As such many of the procedures are somewhat hardwired and rudimentary. If this functionality is to be fully utilized than a more extensive task will be required to automate and streamline the process.

Background

The geometry used for the demonstration is based on the bodies of the fly-back-boosters created in Catia using data from the RAM. Figures 1 and 2 show the before and after top views of the boosters (Actify 3D View used for obtaining views).

Version 4 Catia programming is FORTRAN based with a library of subroutines provided by Catia for geometry manipulation and file management. Documentation for the routines are provided via online access. Although there are samples of subroutines that show how to call the various library routines, it does not provide a sample of an initial main program in which the input files and routines are set up. With the help of Catia technical support (obtained via IBM approximately two weeks after initial help request) the setup was accomplished by adding the proper text to the USRENV.dcls file and calling the initializing routines from the main program. Compiling the code required using the catgeo script included in the Catia administrator's files.

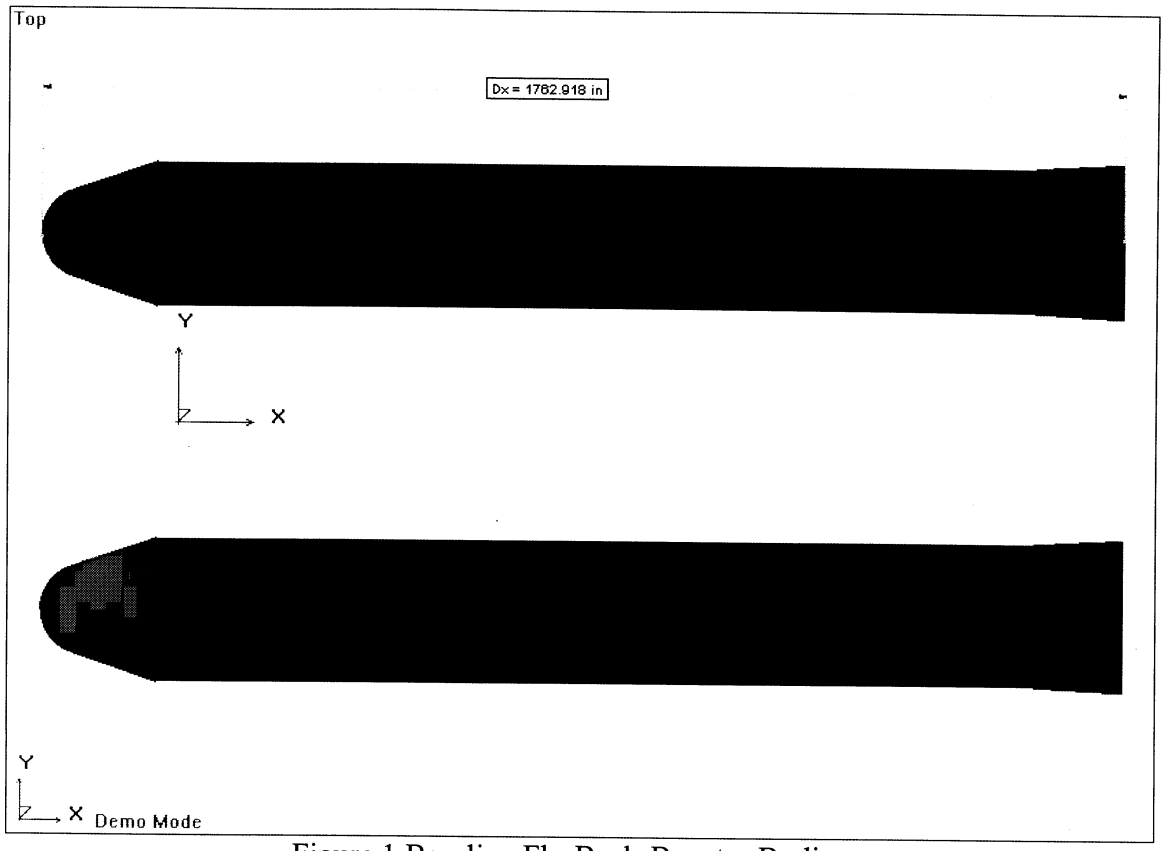


Figure 1 Baseline Fly-Back-Booster Bodies

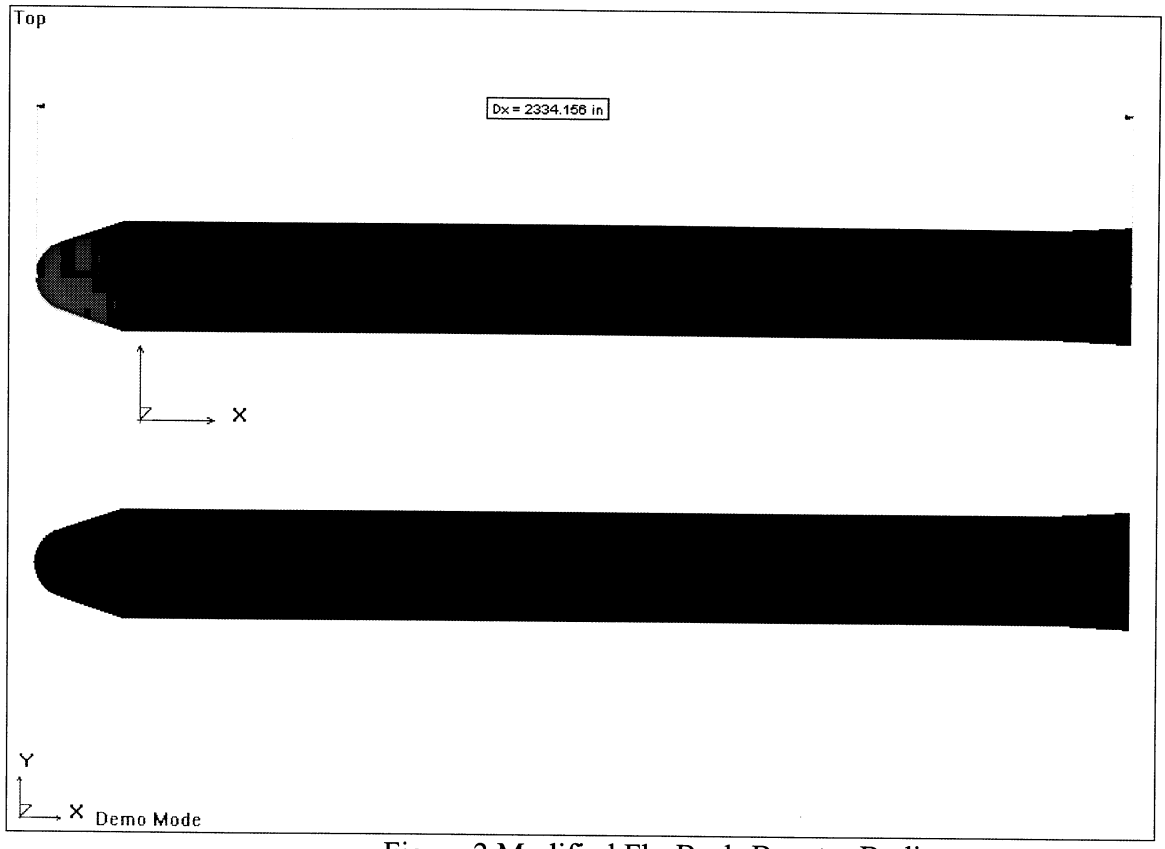


Figure 2 Modified Fly-Back-Booster Bodies

Files

The files used for this task and a brief description of each are listed below. The text of the files listed (except for the Catia models) are included in Appendix A.1. All of the files with the exception of USRENV.dcls (in ~hawke home directory) can be found in Octane02 under the ~hawke/CatiaVMH directory. Appendix B.1 contains the worksheet information referred to at the top of FBBTRAN.f.

- catgeo script used to compile Catia program. Run from a korn shell command line with the following format:
  catgeo –o CATIAPROGRAMNAME
  Note: Catia program name must be in all capitals and usually includes a .f extension not included on the command line.)
- errors.dat contains information about run time errors. An earlier sample is included in Appendix A.
- fbbelement.dat file written after first successful run of FBBTRAN that provides the needed element integer identification for the model entities. This information is used for the rest of the program to identify the desired elements of the geometry. Ultimately a more elegant way of obtaining the needed element information should be possible

using something like a search routine that identifies the desired entities and notes their integer labels for later use in the program.

- FBB_BASELINE_BODIES.model Catia model containing baseline fly-back-booster bodies with known reference points and element names.
- FBB_MOD_BODIES.model Output Catia model containing updated booster body length geometry.
- FBBTRAN.f Catia program run by typing FBBTRAN after compilation using the catgeo script.
- input.dat a one line text file containing the new length of the body.
- USRENV.dcls contains the users Catia environment parameters. This is where the working directory alias is setup. The line added for this task is bolded and underlined in the listing given in Appendix A.1.

Appendix A.1

File Listings

catgeo

```
#!/bin/ksh
#set -x
# PGU   LBY           RISC/6000  HP SGI              COPYRIGHT DASSAULT SYSTEMES 1990
#-------------------------------------------------------------------------------
#                        HOW TO CREATE A CATGEO LOAD
#-------------------------------------------------------------------------------
# CALL    :  catgeo
#-------------------------------------------------------------------------------
# INPUTS  :
#
#     available options :
#     ------------------
#
#   OPTIONAL          -c    : no load is created
#
#   OPTIONAL          -p    : followed by the names of the object programs
#                             specifie to the application
#
#   OPTIONAL          -f    : followed by the file(s) storing the names of
#                             the specific object files to the application
#                     (only AIX options  )
#
#   OPTIONAL          -a    : followed by the name of the librairies containing
#                             the specific object programs to the application
#
#   OPTIONAL          -g    : debug option
#
#   OPTIONAL          -I    : followed by the name of the directory containing
#                             the includes Fortran main files
#                               (There is no -I option on SUN)
#
#   REQUIRED          -o    : load name and main entry name
#
#-------------------------------------------------------------------------------
#
# OUTPUTS :
#             - LINK-EDIT procedure
#             - load (depending on the option used)
#
#
# return code :
# -------------
#             0 : procedure returns normaly
#             2 : option "-o" is missing
#             3 : compilation problem with FORTRAN program
#             5 : parameter file not found
#             7 : bad use of shell.
#
#-------------------------------------------------------------------------------
##############################
#                            #
# RESETTING THE ENVIRONMENT  #
# -------------------------  #
#                            #
##############################
nettoie()
{
  trap ' ' 1 2 15
  if [ $rc -ne 7 ]
  then
      echo "\n             ================================================\n"
      echo "\n                          PROCESS INTERRUPTED  .";
      set `date`
      echo "\n                          Stopped at $4"
      echo "\n             ================================================\n"
  fi
  rm $loadMAJ.o  >/dev/null 2>&1
  exit $rc
```

```
}
trap 'nettoie' 1 2 15
#############################
#                           #
# HOW DOES THE PROGRAM WORK #
# ------------------------- #
#                           #
#############################
useshell()
{
 tput clear
 echo "\t\t==========       catgeo  UTILITY        ==========\n"
 echo "usage :  catgeo [-c] [-p app.o ...] [-f ptofile ...] [-a ptolib ...]"
 echo "                [-g] -o loadname \n"
 echo "options :"
 echo " -c : no load is created "
 echo " -p : followed by the names of the object  programs  specific  to  the"
 echo "      application."
 echo " -a : followed by the name of  the  libraries  containing  the  object"
 echo "      programs specific to the application."
 echo " -g : debug option."
 echo " -I : followed by the name of the directory  containing  the  includes"
 echo "      Fortran files."
 echo " -o : load name and main entry name.\n "
 rc=7
 nettoie
}
###############################################################################
#                          MASTER PROGRAM
###############################################################################
#---------------------
#  USEFULL PARAMETERS
#---------------------
rc=0
# -------------------------
# HOW DOES THE PROGRAM WORK
# -------------------------
if [ $# -lt 2 ]
then
    useshell
fi
# ------------------------------
#  TAKING THE OPTION INTO ACCOUNT
# ------------------------------
Incflag=0
debug=0
loadMaj='no_load'
IncDIR='no_Inc'
os=`uname`
for parameter  in $*
do
  trait=`echo $parameter | awk '/./{ch=substr($0,1,1);print ch}' `
  if [ $trait = "-" ]
  then
      i_option=0
      o_option=0
      case $parameter in
                          -I)  i_option=1
                               Incflag=1 ;;
                          -g)  debug=1 ;;
                          -o)  o_option=1 ;;
                           *)  ;;
      esac
  fi
  if [ $o_option -eq 1 ]                    # option load
  then
      if [ $parameter != "-o" ] && [ $loadMaj = 'no_load' ]
      then
          loadDIR=`dirname $parameter`
```

```
                loadMAJ=`basename $parameter |tr '[a-z]' '[A-Z]'  `
      fi
  fi
  if [ $i_option -eq 1 ]                    # option Include
  then
    if [ $parameter != "-I" ] && [ $IncDIR = 'no_Inc' ]
    then
      ls $parameter > /dev/null 2>&1
      if [ $? -ne 0 ]
      then
           echo "\n\n                 ----> ******** WARNING : ********"
           echo "                       FILE   $parameter    NOT FOUND"
           rc=5
           nettoie
      else
          IncDIR=$parameter
      fi
    fi
  fi
done
# ----------------------------------------
# COMPILING THE MAIN ( FORTRAN, C or C++ )
# ----------------------------------------
if [ -f $loadDIR/$loadMAJ.c ] ; then
        suf=c
        type=c
elif [ -f $loadDIR/$loadMAJ.C ] ; then
        suf=C
        type=cpp
elif [ -f $loadDIR/$loadMAJ.cpp ] ; then
        suf=cpp
        type=cpp
elif [ -f $loadDIR/$loadMAJ.f ] ; then
        suf=f
        type=f
elif [ -f $loadDIR/$loadMAJ.F ] ; then
        suf=F
        type=f
else
        echo " The file " $loadDIR/$loadMAJ " is not a valid catgeo main file "
fi




if [ $type = c ] ; then
   option='-O -c -mips2 -I/usr/lib/ncs/idl/c -I/usr/lib/netls/include \
   -I/usr/lib/ncs/include  -I/usr/include/X11/Xm -dollar '

   compiler=cc

elif [ $type = cpp ]  ; then
   option='-c -no_auto_include -mips2 -xansi -KPIC  -D_LANGUAGE_CPLUSPLUS -D_IRIX_SOURCE -
DIRIX_5_3'
   compiler=CC
else
   option='-static -c -mips2 -Wf,-noappend -O -Nn26000 -Nl1500 -Nq19000 -NC1200 -w1 -Olimit
2000  '

   compiler=f77
fi



   if [ $Incflag -eq 1 ]
   then
      if [ $debug -eq 0 ]
      then
```

```
        $compiler $option -Wf,-I$IncDIR $loadDIR/$loadMAJ.$suf >> compil.res  2>&1
        erreur=$?
      else
        $compiler $option -Wf,-I$IncDIR -g $loadDIR/$loadMAJ.$suf >> compil.res  2>&1
        erreur=$?
      fi
   else
      if [ $debug -eq 0 ]
      then
        $compiler  $option $loadDIR/$loadMAJ.$suf >> compil.res  2>&1
        erreur=$?
      else
        $compiler $option -g  $loadDIR/$loadMAJ.$suf  >> compil.res  2>&1
        erreur=$?
      fi
   fi

if [ $erreur -ne 0 ]
then
    tput clear
    echo "\n"
    echo "             1 - Compile of the main ................ not ok"
    echo "                          ---> consult the compil.res file"
    rc=3
    nettoie
else
    rm compil.res  >/dev/null 2>&1
fi


# ----------------------------------------
# INVOKING THE STANDARD LINK EDIT PROCEDURE
# ----------------------------------------
catlink $* -catgeo
rc=$?
if [ $rc -ne 0 ]
then
   if [ $rc -ne 1 ]
   then
      nettoie
   fi
fi
# ------------------------
# REMOVING TEMPORARY FILES
# ------------------------
rm $loadMAJ.o  >/dev/null 2>&1
# ----------------------------
# BACK TO THE CALLING PROCEDURE
# ----------------------------
exit $rc
```

errors.dat

```
*********************************************************************************
*                  ERROR IN CATGEO/CATMSP SOFTWARE                              *
*          ERROR WHEN USING  SUBROUTINE : GSZGSO                                *
*          CURRENT MODEL NUMBER :     1                                         *
*          ERROR CODE :      2       SEVERITY  4                               *
*    ------------------------------------------------------------------------   *
*   THIS IS NOT AN ELEMENT ADDRESS: 0                                           *
*********************************************************************************

INVALID ADDRESS
```

fbbelement.dat

```
M01
 FBB_BASELINE_BODIES
            1
      I     JADP     NIDEN     JELE    LINDEN     IER
        1        0         5        6      *AXS         0
        2        6         0        7      *AXS         0
        3        7         5        8      *TRA         0
        4        8         5        9      *TRA         0
        5        9         5       11      *AXS         0
        6       11         5       12      *TRA         0
        7       12         4       35      *PT1         0
        8       35         4       36      *PT2         0
        9       36         4       37      *PT6         0
       10       37         4       38      *PT7         0
       11       38         4       39      *PT8         0
       12       39         4       40      *PT9         0
       13       40         4       41      *LN4         0
       14       41         6       43      *CRV         0
       15       43         4       45      *LN5         0
       16       45         4       47      *LN6         0
       17       47         5       49      *LN2         0
       18       49         5       51      *PLN         0
       19       51         5       53      *LN6         0
       20       53         5       55      *SOL         0
       21       55         5       59      *LN7         0
       22       59         5       61      *LN7         0
       23       61         5       63      *SOL         0
       24       63         5       67      *LN9         0
       25       67         5       69      *LN9         0
       26       69         6       71      *SOL         0
       27       71         6       74      *SOL         0
       28       74         6       78      *SOL         0
       29       78         6       82      *SOL         0
       30       82         5       86      *PT1         0
       31       86         5       87      *PT1         0
       32       87         5       88      *PT1         0
 IEND =            1
```

FBBTRAN.f

```
C*************************************************************************
C fbbtran.f
C FORTRAN program to grow length of fly back booster Catia geometry
C*************************************************************************
C Last Revised July 31, 2000
C Veronica M Hawke
C*************************************************************************
C See worksheet to locate points and solids on geometry
C*************************************************************************

      IMPLICIT NONE

      INTEGER*4 IER, JELE, ICOL, MNUM, NBCHAR, ICUR, ISHO, IPIC,
     1          NUMSTK, JPT1, JPT2, JPT, JCPT, JCTR, TRA1,
     2          JSOL, JSO, JGSO, NIDEN, ILAY, JADP, JADF, IEND, I,
     3          IFILT, NBCUT, PT1, PT2,
     4         PT3, PT4, PT5, PT6, PT7, PT8, PT9, PT10, PT11,
     5         SOL1, SOL2, SOL3, SOL4, SOL5, SOL6, SOL7, SOL8,
     6         SOL9, SOL10, SOL11, PLN1,  TRA2, LIDEN, JTR,
     7         JPLN, JLN

      REAL*8 DIST, NEWCYLLEN, DCYLLEN, DX, DY, DZ, PTC1(3),
     1           PTC2(3), RAD, THK

      CHARACTER*8  DirAlias, PROJ, GROUP, USER, ACC, PASSW
      CHARACTER*80 CHAIN, IDENT, TITLE, LAB, TITLE2

      OPEN(UNIT=21, FILE="input.dat")
      OPEN(UNIT=23,FILE="FBB_BASELINE_BODIES.model")
      OPEN(UNIT=24,FILE="fbbelement.dat")
      OPEN(UNIT=25,FILE="FBB_MOD_BODIES.model")
      OPEN(UNIT=26,FILE="errors.dat")


C*************************************************************************
C Set up Catia Geometry Subroutines
C*************************************************************************


      CALL CATGEO

      CALL GUEINI (0,1,26,26,0,0,LAB)

      PROJ = " "
      GROUP = "g10706"
      USER = "hawke"
      ACC = "hawke"
      PASSW = "   "
      IER = 0

      CALL GLOGON (PROJ,GROUP,USER,ACC,PASSW,IER,LAB)

C*************************************************************************
C Open Original Catia Model(s) to be modified
C The directory alias is set to M01 which is defined in USRENV.dcls
C*************************************************************************

      MNUM = 1
      DirAlias = "M01"
      TITLE = "FBB_BASELINE_BODIES"
```

```
        CALL GIMREA ( MNUM, DirAlias, TITLE, 0, 0, IER, LAB)

        CALL GILMDL (MNUM,IER,LAB)

C*************************************************************************
C Get element information from top layer (0)
C*************************************************************************

        WRITE(24,*) DirAlias,TITLE,MNUM
        WRITE(24,*) "     I     JADP    NIDEN    JELE    LINDEN    IER"

        ILAY = 0
        JADP = 0

        DO 10 I = 1, 100

            CALL GISELL ( MNUM, ILAY, JADP, JADF, IEND, IER, LAB)

            IF (IEND.EQ.1) THEN
               GOTO 11
            ENDIF

            JELE = JADF

            CALL GIRIDE (MNUM, JELE, NIDEN, LIDEN, IER, LAB)

            WRITE(24,15) I,JADP,NIDEN,JELE,LIDEN,IER

            JADP = JADF

   10   CONTINUE

15      FORMAT(1X,4I8,2X,A8,I8)

11      WRITE(24,*) "IEND = ",IEND

C*************************************************************************
C Assign integer ID's to elements as found in file created above
C*************************************************************************

        PT1 = 36
        PT2 = 37
        PT3 = 38
        PT4 = 39
        PT5 = 40
        PT6 = 86
        PT7 = 87
        PT8 = 88

        SOL1 = 55
        SOL2 = 63
        SOL3 = 71
        SOL5 = 78
        SOL6 = 74
        SOL7 = 82

        PLN1 = 51

C*************************************************************************
C Delete all old elements except those needed to create new geomtry
C*************************************************************************
```

```
        JELE = SOL5

        CALL GIERAS(MNUM,JELE,IER,LAB)

        JELE = SOL6

        CALL GIERAS(MNUM,JELE,IER,LAB)

        JELE = SOL7

        CALL GIERAS(MNUM,JELE,IER,LAB)

        JELE = SOL2

        CALL GIERAS(MNUM,JELE,IER,LAB)

        CALL GILMDL (MNUM,IER,LAB)
C*************************************************************************
C Create reference point based on new length of body
C*************************************************************************


        READ(21,*) NEWCYLLEN

        JPT1 = PT6
        JPT2 = PT7

        CALL GSOIOO(MNUM,JPT1,JPT2,DIST,IER,LAB)

        DCYLLEN = NEWCYLLEN - DIST

        JPT = PT8
        DX = DCYLLEN
        DY = 0
        DZ = 0

        CALL GSOOCX(MNUM,JPT,DX,DY,DZ,JCPT,IER,LAB)

        PT10 = JCPT

C*************************************************************************
C Create the transformation defined by the translation from PT8 to PT10
C i.e. old aft point on CL to new aft point on CL defined by delta length
C*************************************************************************

        JPT1 = PT8
        JPT2 = PT10

        LIDEN = 24
        IDENT = "Translation along x axis"

        CALL GSTCOO(MNUM,JPT1,JPT2,LIDEN,IDENT,JCTR,IER,LAB)

        TRA1 = JCTR
C*************************************************************************
C Apply TRA1 to aft solid SOL3 creating a new solid SOL8
C Then delete the old solid SOL3
C*************************************************************************

        JSO = SOL3
        JTR = TRA1

        CALL GSZGSO(MNUM,JSO,JTR,JGSO,IER,LAB)
```

```
        SOL8 = JGSO

        JELE = SOL3

        CALL GIERAS(MNUM,JELE,IER,LAB)

C*************************************************************************
C Create new middle cylinder SOL4
C*************************************************************************


        JPT1 = PT3
        JPT2 = PT6

        CALL GSOIOO(MNUM,JPT1,JPT2,DIST,IER,LAB)
        RAD = DIST

        PTC1(1) = -34.17672
        PTC1(2) = 306.0000
        PTC1(3) = -300.0000

        PTC2(1) = PTC1(1) + NEWCYLLEN
        PTC2(2) = PTC1(2)
        PTC2(3) = PTC1(3)


        THK = 0
        NBCUT = 7

        CALL GCWSCY(MNUM,PTC1,PTC2,RAD,THK,NBCUT,JSOL,IER,LAB)

        SOL4 = JSOL

C*************************************************************************
C Define transformation (TRA2) of symmetry about a plane and
C mirror new solids and nose solid across XZ plane of global coordinate
C system (PLN1)
C*************************************************************************

        JPLN = PLN1
        JLN = 0
        LIDEN = 51
        IDENT = "Symmetry about XZ plane of global coordinate system"

        CALL GSTCP1(MNUM,JPLN,JLN,LIDEN,IDENT,JCTR,IER,LAB)

        TRA2 = JCTR

        JTR = TRA2

        JSO = SOL1
        CALL GSZGSO(MNUM,JSO,JTR,JGSO,IER,LAB)
        SOL9 = JGSO

        JSO = SOL4
        CALL GSZGSO(MNUM,JSO,JTR,JGSO,IER,LAB)
        SOL10 = JGSO

        JSO = SOL8
        CALL GSZGSO(MNUM,JSO,JTR,JGSO,IER,LAB)
        SOL11 = JGSO
```

```
        CALL GILMDL (MNUM,IER,LAB)
C*************************************************************************
C Write to new Catia model
C*************************************************************************

        TITLE2 = "FBB_MOD_BODIES"

        CALL GIMWRI(MNUM,DirAlias,TITLE2,1,1,IER,LAB)

        CLOSE(UNIT=21)
        CLOSE(UNIT=22)
        CLOSE(UNIT=23)
        CLOSE(UNIT=24)
        CLOSE(UNIT=25)
        CLOSE(UNIT=26)

      STOP
      END
```

input.dat

```
2000
```

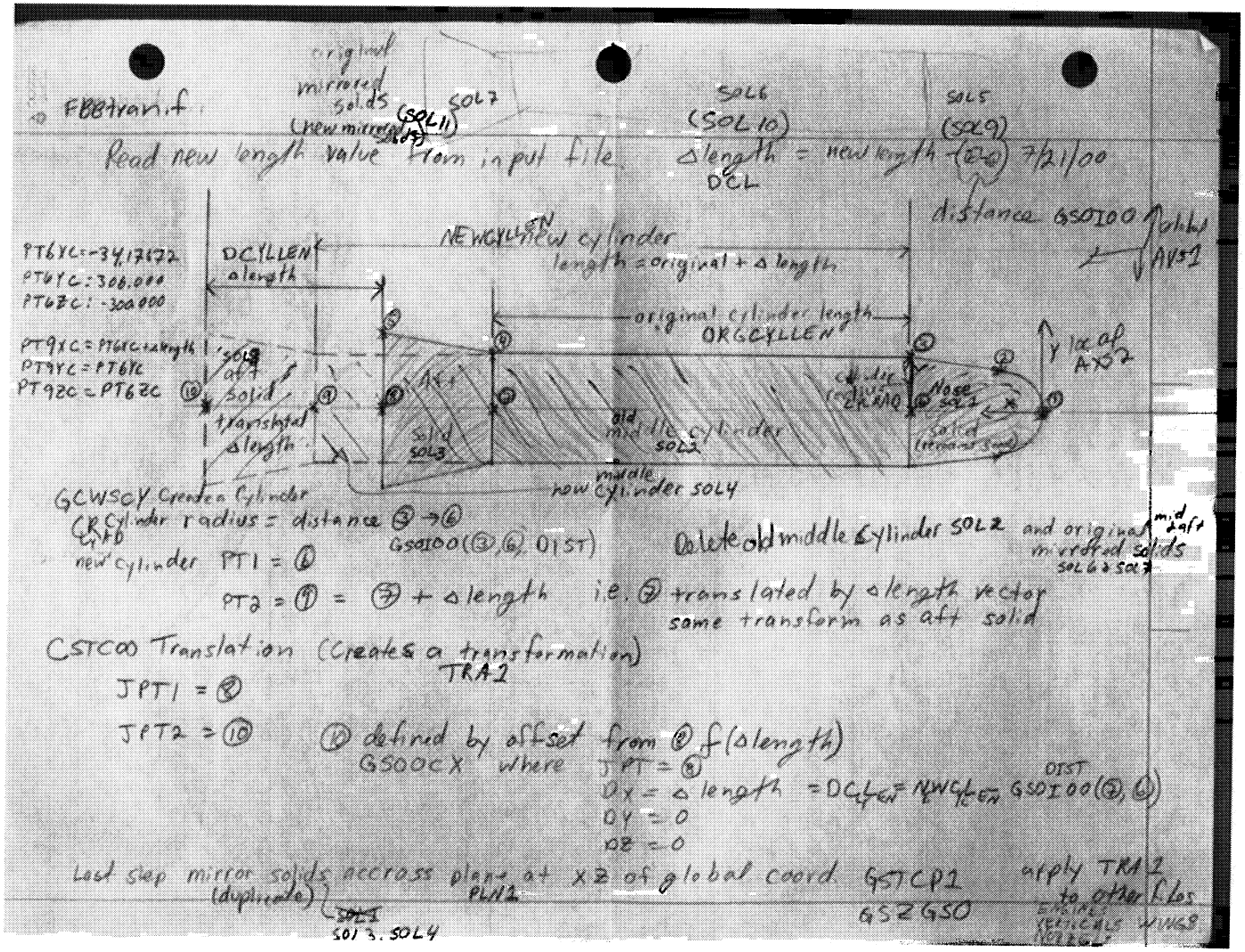
USRENV.dcls

```
/* */
/* ------------------------------------------------------------ */
/* USER RUN TIME DECLARATION FILE                               */
/* ------------------------------------------------------------ */

/* ------------------------------------------------------------ */
/* Number of user interactions between automatic model saves    */
/*  on the ROLL file                                            */
/* [50]                                                         */
/* ------------------------------------------------------------ */

catia.FREQUENCY_ROLL = 50 ;

/* ------------------------------------------------------------ */
/* determines whether the roll file will be re-initialized      */
/* before the CATIA interactive program runs. Therefor          */
/* prohibits the warm start of the catia session                */
/* FALSE                                                        */
/* ------------------------------------------------------------ */
catia.INITIALIZE_ROLL = FALSE ;


/* ------------------------------------------------------------ */
/* Hardcopy of the IBM 5080 graphics screen allowed.            */
/*  With graPHIGS this parameter is ignored.                    */
/* [TRUE]                                                       */
/* ------------------------------------------------------------ */

catia.HARD_COPY_GRAPHIC = TRUE ;

/* ------------------------------------------------------------ */
/* Alphanumeric ANALYSIS window is enabled on alpha screen      */
/* [TRUE]                                                       */
/* ------------------------------------------------------------ */

catia.ANALYSIS_WINDOW = TRUE ;

/* ------------------------------------------------------------ */
/* Length of the graphic screen display list                    */
/*  graPHIGS: number of graphic elements which can be           */
/*            displayed with a multiplication factor of 10.     */
/*            With a value of 400, 4000 graphic primitives      */
/*            can be displayed                                  */
/*  (Deca Graphic Primitives) [400]                             */
/*                                                              */
/*  IBM 5080: size of the local memory of the graphic           */
/*            screen. Also called display list size or          */
/*            graphic buffer size                               */
/* (Kilo Bytes)  400                                            */
/* ------------------------------------------------------------ */

catia.LONBUF_GRAPHIC = 400  ;

/* ------------------------------------------------------------ */
/* Table size in memory which contains references to the        */
/*  graphic segment                                             */
/*  = 2 x LONBUF_GRAPHIC is the recommended value               */
/* (Kilo Bytes)  800                                            */
/* ------------------------------------------------------------ */
```

```
catia.LONMOD_GRAPHIC = 800  ;

/* ------------------------------------------------------------ */
/* Tuning of buffer full in graPHIGS.                           */
/*  Use instead LONBUF_GRAPHIC in GV1/6000.                     */
/*  = (LONBUF_RATIO/1000) x LONBUF_GRAPHIC                      */
/* (Kilo Bytes)  1000                                           */
/* ------------------------------------------------------------ */

catia.LONBUF_RATIO_GRAPHIC = 1000  ;




/* ------------------------------------------------------------ */
/* Default working mode in CATIA at start time.                 */
/*  [SP]                                                        */
/*  [[SP,DR]]                                                   */
/* ------------------------------------------------------------ */

catia.START_MODE = 'SP' ;


/* ------------------------------------------------------------ */
/* Sample MODEL / PICTURE / LIBRARY and SHEET declarations      */
/* ------------------------------------------------------------ */
catia.MODEL    = '$HOME/db';
alias M01 =
catia.MODEL    = '$HOME/CatiaVMH';
catia.PICTURE  = '$HOME/db/picture';
catia.LIBRARY  = '$HOME/db/library';
catia.SHEET    = '$HOME/db';

/* ------------------------------------------------------------ */
/* Include specific declarations for Mechanical or AEC          */
/* ------------------------------------------------------------ */
include( '$HOME/${PREFIX_INDUSTRIAL_DOMAIN}ENV.dcls' ) ;
```

Appendix B.1

Geometry Worksheet


FBBtran.f
original mirrored solids
(new mirrored)
SOL7
SOL6 (SOL10)
SOL5 (SOL9)
Read new length value from input file
Δlength = new length - (DCL)
DCL
7/21/00
distance GSOI00
PT6YC=-34.17672
PT6YC: 306.000
PT6ZC: -300.000
PT9XC = PT6XC + length
PT9YC = PT6YC
PT9ZC = PT6ZC
DCYLLEN
Δlength
NEWCYLLEN
new cylinder length = original + Δlength
original cylinder length
ORGCYLLEN
Y loc of AXS2
SOL8 aft solid translated Δlength
SOL4
SOL3
old middle cylinder SOL2
SOL1 Nose
middle cylinder SOL4
GCWSOY Creates Cylinder
CR Cylinder radius = distance ⑤→⑥
GSOI00(③,⑥, DIST)
Delete old middle cylinder SOL2 and original mirrored solids SOL6 & SOL7
mid aft
new cylinder PT1 = ⑥
PT2 = ⑨ = ⑦ + Δlength i.e. ⑦ translated by Δlength vector
same transform as aft solid
CSTC00 Translation (Creates a transformation)
TRA2
JPT1 = ⑧
JPT2 = ⑩
⑫ defined by offset from ⑧ f(Δlength)
GSOOCX where
JPT = ⑧
DX = Δlength = DCLen - NWCLen
DY = 0
DZ = 0
DIST GSOI00(⑧, ⑫)
Last step mirror solids accross plane at XZ of global coord. GSTCP1
PLN2
apply TRA2 to other SOLs
SOL1 SOL3, SOL4
GS2GSO

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
December 2001

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Summary of ADTT Website Functionality and Features

5. FUNDING NUMBERS
C NAS2-00062
TA 8

6. AUTHOR(S)
Veronica Hawke, Research Scientist,ELORET Cor
Trang Duong, Research Scientist,ELORET Corp.
Lawrence Liang, Research Scientist,ELORET Corp.
Peter Gage, Sr Research Scientist,ELORET Corp.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
ELORET Corp.
690 West Fremont Ave. Suite 8
Sunnyvale, CA 94087-4202

8. PERFORMING ORGANIZATION REPORT NUMBER
AIT 08.01.2000

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/CR 2001-211389

11. SUPPLEMENTARY NOTES
POC: Veronica Hawke ELORET
NASA Ames Research Center, MS 258-1 APS, Moffett Field, CA 94035-1000
(650) 604-4429

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified -- Unlimited
Subject Category: 31, 62, 18 Distribution: Standard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report summarizes development of the ADTT web-based design environment by the ELORET team in 2000. The Advanced Design Technology Testbed had been in development for several years, with demonstration applications restricted to aerodynamic analyses of subsonic aircraft. The key changes achieved this year were improvements in Web-based accessibility, evaluation of collaborative visualization, remote invocation of geometry updates and performance analysis, and application to aerospace system analysis. Significant effort was also devoted to post-processing of data, chiefly through comparison of similar data for alternative vehicle concepts. Such comparison is an essential requirement for designers to make informed choices between alternatives

14. SUBJECT TERMS
Design
Website Applications
Aerospace Vehicle Design
Database Implementation

15. NUMBER OF PAGES
131

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	